Fusion Compiler™
Tool Commands

Version Q-2019.12-SP4, June 2020

SYNOPSYS®

Copyright Notice and Proprietary Information

© 2020 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at
http://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Contents

add_array_to_macro_group

Add macro array to macro group.

SYNTAX

status **add_array_to_macro_group**
-group macro_group
-array macro_array

ARGUMENTS

-macro_group macro_group

Specifies macro group.

-macro_array macro_array

Specifies macro array.

DESCRIPTION

This command add macro array to macro group. All macros of the macro array will be added to the macro group. If the macro array belongs to another macro group before this command, it will be removed from that macro group and added to the new macro group. If the macro group contains macros before the command, those macros and macros of the macro group must belong to the same placeable area, i.e, movebound, voltage area, or block.

EXAMPLES

The following command add macro array array1 to micro group group1

prompt> **add_array_to_macro_group** -group group1 -array array1

SEE ALSO

create_macro_group(2)
remove_array_from_macro_group(2)
add_macro_to_group(2)
remove_macros_from_group(2)
create_macro_array(2)

add_attachment

Adds user defined attachments on a library or a design.

SYNTAX

status **add_attachment**
name
-file_name
-object

Data Types

name string

ARGUMENTS

name

Name of the attachment.

-file_name

Name of the file to be attached.

-object

A library or a design object. A cell library should be used only in Library Manager Shell. lib-cells can not have attachments.

DESCRIPTION

This command adds an attachment to the given library or design. The user attachment is always a file and once attached to the given object, a copy of the file is taken into the database and completely managed by the tool.

EXAMPLES

In the following example an attachment is added to the current block.

prompt> **add_attachment -object [current_block] -file_name test_att.txt test_att**

1

SEE ALSO

remove_attachments(2)
report_attachments(2)
open_attachment(2)

add_buffer

Adds buffer cells on the nets that are connected to the specified pins.

SYNTAX

collection **add_buffer**
[-new_net_names *new_net_names*]
[-new_cell_names *new_cell_names*]
[-inverter_pair]
[-no_of_cells *number*]
object_list
[-lib_cell *buffer_lib_cell*]
[*buffer_lib_cell*]

Data Types

new_net_names list
new_cell_names list
number integer
object_list list
buffer_lib_cell collection

ARGUMENTS

-new_net_names *new_net_names*

Specifies the names of the new nets to add. You must specify one net name per buffer when adding buffers, and two net names per inverter pair when adding inverter pairs. If the specified net name already exists, the command adds a suffix of "_%d" or "_%d_%d" to the net name.

Optionally, if you specify only the common base name, the tool generates new net names by adding unique numeric suffixes to the common base name. The specified names can be any valid net names, but must be the leaf names. They must not be the hierarchical names and must not contain embedded hierarchical separators. They must be unique in the current context, as specified by the current instance. By default, the command uses the base name eco_net.

-new_cell_names *new_cell_names*

Specifies the names of the new cells to be added. You must specify one cell name per buffer when adding buffers, and two cell names per inverter pair when adding inverter pairs. If the specified cell name already exists, the command adds a suffix of "_%d" or "_%d_%d" to the cell name.

Optionally, if you specify only the common base name, the tool generates new cell names by adding unique numeric suffixes to the common base name. These names can be any valid cell names, but must be the leaf names. They must not be the hierarchical names and must not contain embedded hierarchical separators. They must be unique in the current context, as specified by the current instance.

By default, the command uses *eco_cell* as the common base name.

-inverter_pair

Adds inverter pairs instead of buffer cells. If you specify this option, you must supply a library cell that has an inverting output. You can use this option when the specified library cell or buffer has both inverting and noninverting outputs.

-no_of_cells ***number***

Specifies the number of buffer cells or inverter pairs to be inserted per net. The inserted repeaters are connected back-to-back in series. By default, the command inserts a single buffer cell or inverter pair per net.

object_list

Specifies a list of nets, pins, or ports that must be buffered. The new buffer cells or inverter pairs are placed close to the specified pins or ports if their cells are placed.

If you specify a net, the tool connects the buffers or inverter pairs such that a new buffer cell is the new load of the original net.

If you specify pins, the tool groups all of the specified pins based on the nets to which they are connected. When the grouped pins are load pins, the tool adds the buffers so that the new buffer cells can drive them. When the grouped pins are driver pins, the tool connects the new buffer cells so that they become the load of the specified driver pin.

-lib_cell ***buffer_lib_cell***

Specifies the library cell to be used as buffer. In this case, the object is either a named library cell or a library cell collection. This is a required option.

If the library cell has both inverting and noninverting outputs, that is, it can act both as buffer and inverter, the **-inverter_pair** option controls which output is used. If the library cell has multiple outputs, the command uses the first noninverting or inverting output.

buffer_lib_cell

This positional option is exactly equivalent to -lib_cell. And it is mutual exclusive with **-lib_cell**.

DESCRIPTION

This command adds buffers or inverter pairs at one or more specified pins or ports. A library cell with a single input and multiple outputs is a buffer, as long as each output has the same or inverted logic function as the input.

Like all other netlist editing commands, the command arguments to **add_buffer** must be valid for the command to run successfully. If the command succeeds, the result is a collection of the newly-added cells. If the command fails, the command returns an empty collection or an empty string and does not change the netlist.

By default, each newly-created cell has a name beginning with the eco_cell string and ending with a unique numeric suffix. Each newly-created net has a name beginning with the eco_net string and ending with a unique numeric suffix. To override the automatic name generation by the tool, use the **-new_net_names** and **-new_cell_names** options.

You can mimic buffer addition by using other commands, such as **create_cell**, **create_net**, **disconnect_net**, and **connect_net**. The **add_buffer** command provides a more efficient and safe way to add buffers.

This command support editing a Verilog module in the module aspect, through **edit_module**.

This command will check the editability of the block before adding buffer cells.

This command honors the **design.eco_freeze_silicon_mode** application option. When this application option is **true**, the

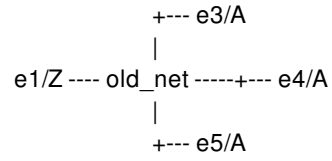
is_fs_eco_add attribute of the added buffers is set to **true**.

The **add_buffer** command uses the following basic rules to check its arguments for validity:

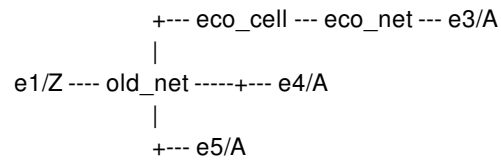
- The pin or port must be in scope; that is, at or below the current instance. For a description of special scoping rules, see the "Buffering Inside Boundary Pins" section.
- The pin or port must be connected to a net.
- Bidirectional pins cannot be buffered.
- The net cannot be either a PG net or a tie net.
- The specified library cell cannot be sequential.
- The specified library cell must be a buffer, as previously defined.
- The *number* argument of the **-no_of_cells** option must be a positive, nonzero integer.
- The library cell must have an inverting output, when you use the **-inverter_pair** option. The command uses the first inverting output and adds two cells so that it preserves the logic of the path.

Adding and Connecting the New Buffer

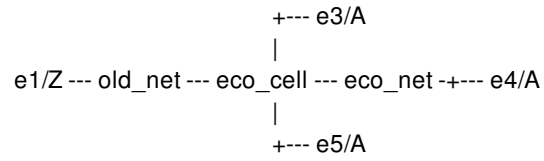
The following figure shows an example network that is used to describe the rules used by the **add_buffer** command to connect the new buffer or inverter pair:



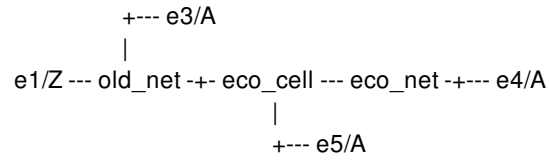
If the specified pin is a load, the load is first disconnected from its old net and then connected to the new net, as shown in the following figure:



If the specified pin is a driver, all loads on that net are disconnected from the old net and are connected to the new net, as shown in the following figure:



If you specify a list of load pins on the same net, these load pins are grouped and the new net drives them, as shown in the following figure:



Note that the tool cannot buffer a net driven by multiple drivers.

Buffering Inside Boundary Pins

When you use this command to add a buffer at a pin on the boundary of a hierarchical block, the command adds the buffer either inside or outside the hierarchical block, depending on the current hierarchical scope. To add the buffer inside the hierarchical block, set the scope to that block by using the **current_instance** command, and then run the **add_buffer** command. The tool adds the buffer within the block to which the **current_instance** is set.

Don't touch support for nets

When app option *eco.add_buffer.honor_dont_touch* is set to true, nets with dont_touch attribute are skipped.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies a library cell that is not a buffer. The command fails and generates an error message.

```
prompt> add_buffer e1/Z -lib_cell class/AN2P
Information: Buffering net old_net. (HFS-701)
Error: library cell 'AN2P' is not a buffer or an inverter. (ECOUI-010)
```

The following example specifies a buffer for the library cell:

```
prompt> add_buffer e1/Z -lib_cell class/B1I
{eco_cell}
```

The following example specifies an inverter for the library cell. The command succeeds and creates the new cells named *eco_cell* and *eco_cell_1*. The new nets are named *eco_net* and *eco_net_1*. The **report_cells** command displays the connections for cell e3.

```
prompt> add_buffer e3/A -lib_cell class/IV -inverter_pair
{eco_cell eco_cell_1}

prompt> report_cells -connections e3
****************************************
Report : cell
 -connections
****************************************

Connections for cell 'e3':
 Reference:   B1I
 Library:     class

 Input Pins        Net
 ----------------  ------------
 A                 eco_net

 Output Pins       Net
 ----------------  ------------
 Z                 out2
```

```
1
prompt> report_nets -connections eco_net
****************************************
Report : net
 -connections
****************************************

Connections for net 'eco_net':

 Driver Pins      Type
 ------------     ----------------
 eco_cell/Z       Output Pin (IV)

 Load Pins        Type
 ------------     ----------------
 e3/A             Input Pin (B1I)

1
```

Editing a Verilog module in the module aspect:

```
prompt> edit_module r4000 { add_buffer -lib_cell BUFFHVTD2 MemWrite \
  -new_cell_names ecoCell -new_net_names ecoNet}
```

SEE ALSO

current_instance(2)
edit_module(2)
get_pins(2)
remove_buffers(2)
report_cells(2)
report_nets(2)
size_cell(2)
eco.add_buffer.honor_dont_touch(3)

add_buffer_on_route

Adds buffers along the route of the specified nets.

SYNTAX

add_buffer_on_route
list_of_nets
[-lib_cell *buffer_lib_cell* | *buffer_lib_cell*]
[-net_prefix *prefix*]
[-cell_prefix *prefix*]
[-location *coordinate_list*]
[-user_specified_buffers *user_specified_buffer_list*]
[-max_distance_to_route *length*]
[-detect_layer]
[-user_specified_bus_buffers *user_specified_bus_buffer_list*]
[-bus_cell_prefix *bus_cell_prefix_list*]
[-repeater_distance *length* | -repeater_distance_length_ratio *length_ratio*
[-first_distance *length* | -first_distance_length_ratio *length_ratio*]
[-scaled_by_layer *layer_scale_list*]
[-scaled_by_width]
[-only_global_routed_nets]
[-respect_blockages [-allow_insertion_over_cell *cell_list*]
| [-dont_allow_insertion_over_cell *cell_list*]]
[-max_distance_for_incomplete_route *length*]
[-snap_to_sites]
[-punch_port]
[-respect_gas_station *supply_net*]
[-max_distance_route_to_gas_station *length*]
[-respect_voltage_areas]
[-voltage_area_specific_lib_cells *voltage_area_specific_lib_cell_list*]
[-allow_physical_feedthrough_buffer *voltage_areas*]
[-on_top_hierarchy]
[-max_distance_to_spare_cell *length* | -not_spare_cell_aware]
[-select_mv_buffers *mv_aware_lib_cell_list*]
[-verbose]

Data Types

list_of_nets list
buffer_lib_cell string
prefix string
coordinate_list list
user_specified_buffer_list list
length float
user_specified_bus_buffer_list list
bus_cell_prefix_list list
length_ratio float

layer_scale_list list
cell_list list
supply_net string
voltage_area_specific_lib_cell_list list
voltage_areas collection
mv_aware_lib_cell_list list

ARGUMENTS

list_of_nets

Specifies the nets to be buffered. If **-user_specified_buffers** or **-location** is used, this option must specify only a single net name.

-lib_cell ***buffer_lib_cell***

Specifies the library cell object to use as a repeater. The object is either a named library cell or a library cell collection.

The library cell must be either buffer or inverter. The command automatically determines the type of the library cell and inserts buffer or inverter pairs on routes based on the type.

This option cannot be used with the **-user_specified_buffers** or **-user_specified_bus_buffers** options. If **-location**, **-repeater_distance** or **-repeater_distance_length_ratio** is specified, this option must also be specified.

buffer_lib_cell

This positional option is exactly equivalent to the **-lib_cell** option. This option is mutually exclusive with the **-lib_cell** option.

-net_prefix ***prefix***

Specifies the prefix to be used for new nets that are created when buffers are inserted in the netlist. By default, the command uses "eco_net" as the prefix.

-cell_prefix ***prefix***

Specifies the prefix to be used for new buffers that are inserted in the netlist. By default, the command uses "eco_cell" as the prefix. This option cannot be used with the **-user_specified_buffers** option.

-location ***coordinate_list***

Specifies the center point coordinates for the locations of the buffer or inverter pair cells. The format is:

{*x1 y1 layer1 x2 y2 layer2* ...}

You must specify one location per buffer when inserting buffers. You must specify two locations per inverter pair when inserting inverter pairs. These locations can be any valid {x_coordinate y_coordinate layer_name} triplet that is not far away from the placement region of the core area and is relative to the chip origin. These locations must be within a region from the route. The region radius is specified by the **-max_distance_to_route** option. Specify the coordinates in user units. More than one routing shape can overlap the specified location. The layer is used to decide which shape to pick.

- If you specify a valid name for the layer, the location is matched on a routing shape on the specified layer.
- If you specify the **-detect_layer** option, the coordinate_list must use the format {x1 y1 x2 y2}, with no layer name specification. The command tries to find the nearest routing shape. If there are two routing shapes in different metal layers that are the same distance from the specified location, the command writes out an error message and exits.

For best results, specify the explicit layer instead of using the **-detect_layer** option.

-user_specified_buffers ***user_specified_buffer_list***

Specifies the center point coordinates for the locations of the buffer or inverter pair cells, as well as their names and library cells. The format is:

{*name1 lib_cell1 x1 y1 layer_name1 name2 lib_cell2 x2 y2 layer_name2* ...}

You must specify one five-tuple per buffer when inserting buffers. You must specify two five-tuples per inverter pair when inserting inverter pairs. The names should be valid name strings for buffers or inverters. The library cells should be name strings or collections. They can be all inverters, all buffers or buffers/inverters mixed. When input mixed buffers/inverters, the inverters must be pairs in each branch path. The locations can be any valid coordinates with layers that are not far away from the placement region of the core area and are relative to the chip origin. These locations must be within a region from the route. The region is specified by **-max_distance_to_route**. Specify the coordinates in user units. More than one routing shape can overlap the specified location. The layer is used to decide which shape to pick. This option cannot be used with the **-lib_cell** or **-voltage_area_specific_lib_cells** options.

- If you specify a valid name for the layer, the location is matched on a routing shape on the specified layer.
- If you specify the **-detect_layer** option, the user_specified_buffer_list must use the format { name1 lib_cell1 x1 y1 name2 lib_cell2 x2 y2 }, with no layer name specification. The command tries to find the nearest routing shape. If there are two routing shapes in different metal layers that are the same distance from the specified location, the command writes out an error message and exits.

For best results, specify the explicit layer instead of using the **-detect_layer** option.

-max_distance_to_route ***length***

Specifies the maximum horizontal or vertical distance in user units between the specified location and route. The default for this option is five times the height of reference cell. You can use this option only if you also use the **-location** or **-user_specified_buffers** option.

-detect_layer

Specifies that the tool detects the layer of location coordinates automatically. The tool tries to find the nearest routing shape. If there are two routing shapes in different metal layers that are the same distance from the specified location, the tool will error out. When this option is used, the coordinate_list uses the format {x1 y1 x2 y2} or {name1 lib_cell1 x1 y1 name2 lib_cell2 x2 y2}, no layer name specified.

You can use this option only if you also use the **-location** or **-user_specified_buffers** option.

-user_specified_bus_buffers ***user_specified_bus_buffer_list***

Specifies the center point coordinates for the locations of bus buffer or inverter pair cells, as well as their names and library cells, using the format {*bus_buffer_pattern1 lib_cell1 x1 y1 bus_buffer_pattern1 lib_cell2 x2 y2* ...}. You must specify one four-tuple per buffer added on each bit at a location. You must specify two four-tuples per inverter pair when inserting inverter pairs. The *bus_buffer_patterns* should be name strings or collections created by the **create_eco_bus_buffer_pattern** command. The library cells should be name strings or collections. They must be either all inverters or all buffers. The locations can be any valid coordinates that are inside the placement region of the core area and are relative to the chip origin. These locations must be within routes of the bus. Routes of all bits near this point must be parallel and match style (horizontal or vertical) of the specified pattern. For horizontal routes, the center of the bus buffer pattern is located at the x of the specified location. For vertical routes, the center of the bus buffer pattern is located at the y of the specified location. And routes of all bits near this point must be in same situation (e.g. same VA, not over blockage/macro). Specify the coordinates in units of user unit.

This option cannot be used with the **-repeater_distance**, **-repeater_distance_length_ratio**, **-location** or **-user_specified_buffers** options. This option cannot be used with the **-lib_cell** or **-voltage_area_specific_lib_cells** options.

-bus_cell_prefix ***bus_cell_prefix_list***

Specifies the prefix to be used for new buffers that are added by the **-user_specified_bus_buffers** option.

If this option not specified, or the pattern of bus buffers are not listed in *bus_cell_prefix_list*, tool uses the prefix specified by *-cell_prefix*. If *-cell_prefix* is also not specified, default prefix will be name of the bus buffer pattern

You can use this option together with the *-user_specified_bus_buffers* option.

-repeater_distance *length*

Specifies the distance in user units between each buffer to be inserted along the route. This option applies to the entire physical route for the specified nets.

-repeater_distance_length_ratio *length_ratio*

Specifies the distance between each buffer that is inserted along the route as a ratio of the total route length. The value range is between 0.0 and 1.0. This option applies to the entire physical route for the specified nets.

Only one of the **-location**, **-user_specified_buffers**, **-user_specified_bus_buffers**, **-repeater_distance**, or **-repeater_distance_length_ratio** options should be specified.

-first_distance *length*

Specifies the distance in user units between the driver pin and the first buffer introduced along the route. This option must be used together with the **-repeater_distance** or **-repeater_distance_length_ratio** option.

-first_distance_length_ratio *length_ratio*

Specifies the distance between the driver pin and the first buffer introduced along the route as a ratio of the total route length. The value range is between 0.0 and 1.0. This option must be used together with the **-repeater_distance_length_ratio** option.

-scaled_by_layer *layer_scale_list*

Scales the distance between buffers by the scaling factor specified for the layer. Specify the scaling factors by using the following format:

{{*layer1 factor1*} {*layer2 factor2*} ...}

The command scales the distance on each layer with the following formula:

D = d*L_M

d:the repeater distance
D:the repeater scaled distance
L_M:layer scaling factor on metal M

Layers that are not in the list use a default factor of 1. You must specify this option together with the **-repeater_distance** option.

-scaled_by_width

Scales the distance between buffers by the ratio of the default width for the layer and the actual route width. This option is mutually exclusive with the **-only_global_routed_nets** option.

The command scales the distance on each layer with the following formula:

D = d*w_M/w_R

d:the repeater distance
D:the repeater scaled distance
w_M:default route width on metal M
w_R:actual route width

By default, the scaling factor is 1 (w_M=w_R). You must specify this option together with the **-repeater_distance** option.

-only_global_routed_nets

Enable buffer insertion on global routed nets. When this option is used, the specified nets must be only global routed without track assignments or detail routes. This option is mutually exclusive with the **-scaled_by_width** option.

-respect_blockages

Specifies that no buffer should be added within a placement blockage, soft macro, or hard macro.

-allow_insertion_over_cell *cell_list*

Specifies the soft macro or hard macro instances in which the repeaters can be added. This option must be used together with the **-respect_blockages** option.

-dont_allow_insertion_over_cell *cell_list*

Specifies the soft macro or hard macro instances in which the repeaters are not allowed. This option is mutually exclusive with the **-respect_blockages** option.

-max_distance_for_incomplete_route *length*

Defines the maximum search distance for incomplete routes. By default, the search distance is one repeater height.

-snap_to_sites

Snaps the newly added buffers to the nearest sites.

-punch_port

Enables buffer insertion on a route segment, even when the topology is different than the netlist connection in the logical hierarchy. In this situation, the tool creates a hierarchical port to connect the loads of the buffer. This option is mutually exclusive with the **-respect_voltage_areas**, **-voltage_area_specific_lib_cells** and **-respect_gas_station** options.

-respect_gas_station *supply_net*

Buffers will only be added within gas stations (whose is_primary attribute is false) for the specified *supply_net* and along the routes. The buffers that are inserted will belong to the gas station. This option automatically enables the **-respect_voltage_areas** option. This option is mutually exclusive with the **-punch_port** option.

-max_distance_route_to_gas_station *length*

Specifies the distance between the gas station in which the buffer is added the route. If this option is not specified, gas stations must be on the route.

-respect_voltage_areas

Inserts buffers within the hierarchy of the voltage area such that the buffer is physically inside hierarchy in the layout. When this option is specified, buffers cannot be added outside core area. This option is mutually exclusive with the **-punch_port** option.

-voltage_area_specific_lib_cells *voltage_area_specific_lib_cell_list*

Specifies library cell objects to be used in the specified voltage areas. The format is:

{*va1 lib_cell1 va2 lib_cell2* ...}

For the default voltage area, specify DEFAULT_VA as the voltage area name. If a buffer is added within a voltage area that is not specified by this option, the lib cell specified by **-lib_cell** will be used. When the **-voltage_area_specific_lib_cells** is specified, buffers cannot be added outside core area. The **-lib_cell** option is always required when **-voltage_area_specific_lib_cells** is specified. This option will automatically enable the **-respect_voltage_areas** option. This option is mutually exclusive with the **-user_specified_buffers** and **-punch_port** options.

-allow_physical_feedthrough_buffer *voltage_areas*

Specifies voltage areas in which buffers can be added, even if they are physical-only feedthroughs. *voltage_areas* must by a

primary voltage area. This option must be used together with the **-respect_voltage_areas** or **-voltage_area_specific_lib_cells** option.

-on_top_hierarchy

Adds buffers to the net at the highest level of hierarchy. By default, the buffers are added in the lowest common ancestor hierarchy of the pins driven by the buffers.

-max_distance_to_spare_cell ***length***

Specifies the maximum distance between newly added buffers and spare cells in freeze-silicon mode. If this option is not specified, the command uses the default of 5 site heights. For user location based, buffer is inserted at the projected location on the route; there must be a compatible spare cell within the maximum search distance or the command will write out an error message and exit. For repeater distance based, the buffer is added only when there is a compatible spare cell within the maximum search distance. The newly added buffers are placed to overlap the matched spare cells and a new "fs_mapped_cell_name" attribute is added to record the mapping relationship in both buffers and spare cells. The feature of spare cell aware only works in freeze-silicon mode, inverter pair input and some other advanced options. This option is mutually exclusive with the **-not_spare_cell_aware** option.

-not_spare_cell_aware

Disables the spare cell-aware feature in freeze-silicon mode. By default, this option is off. The spare cell-aware feature only works in freeze-silicon mode. This option is mutually exclusive with the **-max_distance_to_spare_cell** option.

-select_mv_buffers ***mv_aware_lib_cell_list***

Specifies single rail and dual rail library cell objects to be used in the specified voltage areas. The format is:

{*va1 single_rail_lib_cell1 dual_rail_lib_cell1 va2 single_rail_lib_cell2 dual_rail_lib_cell2* ...}

When this option is specified, command will enable mv aware features. It will select the mv legal lib cell from the input list to create buffers and update supply net automatically. For the default voltage area, specify DEFAULT_VA as the voltage area name. For the voltage areas not specified by this option, command will use the lib cell specified by **-lib_cell**. This option will automatically enable the **-respect_voltage_areas** option.

-verbose

Shows verbose messages.

DESCRIPTION

This command adds buffers along the route of the specified nets. The new buffers are placed but not legalized, and the original net and the newly created nets are not rerouted.

The new buffers are placed close to the route of the input net. The new nets follow the original route as much as possible. User can query the new buffers by the collection *abor_new_buffers*.

A library cell with a single input and multiple outputs is a buffer, as long as each output has the same or inverted logic function as the input.

By default, each newly added buffer has a name beginning with the prefix eco_cell and ending with a numeric suffix. To override the automatic name generation by the tool, use the **-net_prefix** and **-cell_prefix** options.

This command also supports spare cell aware feature in freeze-silicon mode.

The **add_buffer_on_route** command uses the following basic rules to validate the command arguments:

- The net cannot be a PG net or a tie net.
- The net can be routed incompletely. But all routing segments must be fully interconnected and there is one and only one wire within search distance from each untouched load pin. The search distance can be specified by the **-max_distance_for_incomplete_route** option.
- The library cell cannot be sequential.
- The library cell must be a buffer or inverter, as previously defined.

This command honors the **design.eco_freeze_silicon_mode** application option. When this application option is true, the is_fs_eco_add attribute is set to true for the inserted buffers.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the command adds buffers on net1 using the lib/BUF library cell. The repeater distance is 100 and the first distance is 80.

```
prompt> add_buffer_on_route -repeater_distance 100 \
  -first_distance 80 net1 -lib_cell lib/BUF
```

The following example adds buffers on net1 using the lib/BUF library cell. One location is specified and let command to detect layer.

```
prompt> add_buffer_on_route -location {100 200} -detect_layer net1 \
  -lib_cell lib/BUF
```

The following example adds buffers on net1 using the lib/BUF library cell. Both location and layer are specified.

```
prompt> add_buffer_on_route -location {100 200 METAL3} net1 \
  -lib_cell lib/BUF
```

The following example adds buffers on net1 by using the **-user_specified_buffers** option.

```
prompt> add_buffer_on_route n1 \
  -user_specified_buffers {eco1 bufbd2 70 10 eco2 bufbd4 140 10} \
  -detect_layer
```

The following example adds buffers on bus data* by using the **-user_specified_bus_buffers** option.

```
prompt> add_buffer_on_route [get_nets data*] \
  -user_specified_bus_buffers "bpuLT bufx8 x1 y1 bpuBR bufx8 x2 y2"
```

The following example adds buffers on net1 using the lib/BUF library cell. Repeater distance is 40 percent of the total route length and a first distance of 20 percent of the total route length is specified.

```
prompt> add_buffer_on_route \
  -repeater_distance_length_ratio 0.4 \
  -first_distance_length_ratio 0.2 \
  net1 -lib_cell lib/BUF
```

The following example adds buffers on net1 using the lib/BUF library cell. A repeater distance of 100 is used. The distance scaling factor for METAL2 is 0.9 and the distance scaling factor for METAL3 is 0.8. The repeat distance is scaled by the route width on each layer.

```
prompt> add_buffer_on_route -repeater_distance 100 \
  -scaled_by_width -scaled_by_layer { {METAL2 0.9} {METAL3 0.8} } \
  net1 -lib_cell lib/BUF
```

The following example snaps the new added buffers to the nearest sites.

```
prompt> add_buffer_on_route -location {100 200 METAL3} net1 \
  -lib_cell lib/BUF -snap_to_sites
```

The following example creates the necessary hierarchical ports after buffer insertion.

```
prompt> add_buffer_on_route -repeater_distance 100 h1/net1 \
  -lib_cell lib/BUF -punch_port
```

The following example inserts buffers while respecting voltage areas.

```
prompt> add_buffer_on_route -repeater_distance 100 h1/net1 \
  -lib_cell lib/BUF -respect_voltage_areas
```

The following example respects voltage areas and adds buffer in physical-only feedthrough voltage areas.

```
prompt> add_buffer_on_route -repeater_distance 100 h1/net1 \
  -lib_cell lib/BUF -respect_voltage_areas \
  -allow_physical_feedthrough_buffer {VA1 VA2}
```

The following example specifies lib cells for voltage areas. Buffers added in va1 will use lib cell BUF1; buffers added in default voltage area will use lib cell buf2; buffers added in all other voltage areas will use lib cell BUF0.

```
prompt> add_buffer_on_route -repeater_distance 100 net1 \
  -lib_cell lib/BUF0 \
  -voltage_area_specific_lib_cells {va1 lib/BUF1 DEFAULT_VA lib/BUF2}
```

The following example adds buffers only within 10 user unit of gas stations for net vdd1. The buffers will belong to those gas stations.

```
prompt> add_buffer_on_route -repeater_distance 100 net1 \
  -lib_cell lib/BUF -respect_gas_station vdd1 \
  -max_distance_route_to_gas_station 10
```

The following example shows enable mv aware features in add_buffer_on_route.

```
prompt> add_buffer_on_route -repeater_distance 100 net1 \
  -lib_cell lib/BUF \
  -select_mv_buffers {VA1 lib/BUFS1 lib/BUFD1 VA2 lib/BUFS2 lib/BUFD2}
```

SEE ALSO

create_eco_bus_buffer_pattern(2)
legalize_placement(2)
place_eco_cells(2)
remove_buffers(2)
design.eco_freeze_silicon_mode(3)

add_command_hook

Add hook into command execution

SYNTAX

string **add_command_hook** -before *script*
-after *script*
-replace *script*
[-name *name*]
commandName

string *script*
string *script*
string *script*
string *name*
string *commandName*

ARGUMENTS

-before *script*

Hook runs before command.

-after *script*

Hook runs after command.

-replace *script*

Hook runs in place of command.

-name *name*

Name of hook. If no name is specified a name is automatically generated.

commandName

Command to update.

DESCRIPTION

The **add_command_hook** adds a customization point into the execution of an application command. For example, these hooks can

be used to either run a report or gather statistics before and after an application command is run.

Within the body of any hook the get_current_hook_command can be used to query the command and its arguments. For example, a before and after hook may use this information if the hook actions should only happen when specific options are specified. In a replace hook the hook can run the replaced command possibly updating the command options.

This command returns the name of the hook which can be used to remove the hook from the command if desired.

EXAMPLES

Arrange for report timing to be executed every time place_opt_is run.

```
prompt> add_command_hook place_opt -before report_timing
before0
```

Force all report_timing calls to use the -nosplit option.

```
prompt> add_command_hook report_timing -replace {eval [get_current_hook_command] -nosplit}
replace0
```

SEE ALSO

get_command_hooks(2)
remove_command_hook(2)
get_current_hook_command(2)

add_eco_repeater

Adds buffer cells.

SYNTAX

collection **add_eco_repeater**
-cell_name *repeater_name*
-lib_cell *buffer_lib_cell*
-input_net_name *input_net_name*
-output_net_name *output_net_name*
[-load *list_of_pin_or_port_names*]

Data Types

repeater_name	string
buffer_lib_cell	collection
input_net_name	string
output_net_name	string
list_of_pin_or_port_names	list

ARGUMENTS

-cell_name *repeater_name*

Specifies the name of the repeater to add. It should not be same with the name of an existing cell.

-lib_cell *buffer_lib_cell*

Specifies the library cell to be used as buffer. In this case, the object is either a named library cell or a library cell collection. It must be a buffer or inverter.

-input_net_name *input_net_name*

Specifies the names of the input net. If it is an existing net, it will be connected to the input pin of the repeater; otherwise, a new net will be created as this name and connected to the input pin of the repeater. The net cannot be either a PG net or a tie net.

-output_net_name *output_net_name*

Specifies the names of the output net. If it is an existing net, it will be connected to the output pin of the repeater; otherwise, a new net will be created as this name and connected to the output pin of the repeater. The net cannot be either a PG net or a tie net.

-load *list_of_pin_or_port_names*

The specified loads will be driven by the added repeater. Such pins or ports must be connected with the specified and existing input_net or output_net If this option is not specified, all loads of the existing input_net or output_net will be driven by the added repeater.

DESCRIPTION

This command inserts a repeater in an existing net, to buffer all or part of its loads. You must specify such net as either *input_net_name* or *output_net_name*, while the other net must be specified as a new net.

This command support editing a verilog module in the module aspect, through **edit_module**.

This command will check the editability of the block before adding buffer cells.

This command honors the **design.eco_freeze_silicon_mode** application option. When this application option is **true**, the **is_fs_eco_add** attribute of the added buffers added is set to **true**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies the existing net n1 as *input_net_name* and all of its loads will be buffered.

```
prompt> add_eco_repeater -cell_name ecoCell -input_net_name n1 \
   -output_net_name newNetName -lib_cell INVLVTD0
```

The following example specifies the existing net n1 as *output_net_name* and all of its loads will be buffered.

```
prompt> add_eco_repeater -cell_name ecoCell -input_net_name newNetName \
   -output_net_name n1 -lib_cell INVLVTD0
```

The following example specifies the existing net n1 as *output_net_name* and only its loads {U1/A U2/A} will be buffered, other loads connected with n1 will not be buffered.

```
prompt> add_eco_repeater -cell_name ecoCell -input_net_name newNetName \
   -output_net_name n1 -lib_cell INVLVTD0 -load {U1/A U2/A}
```

The following example shows that this command also support editing a verilog module.

```
prompt> edit_module h1m {add_eco_repeater -cell_name ecoCell \
   -input_net_name newNetName -output_net_name n1 -lib_cell \
   INVLVTD0 -load {U1/A U2/A} }
```

SEE ALSO

edit_module(2)
remove_eco_repeater(2)

add_feedthrough_buffers

Inserts buffers on all feedthrough nets.

SYNTAX

status **add_feedthrough_buffers**
[-type buffer | inverter]
[-buffer_side input | output | both]
[-user_buffer *cell_ref*]
[-nets *nets*]
[-logical]
[-verbose]

Data Types

cell_ref string
nets collection

ARGUMENTS

-type buffer | **inverter**

Specifies the type of cell, buffer or inverter, to use for isolating feedthroughs. By default, a buffer is used.

-buffer_side input | **output** | **both**

Specifies what side of the net should the feedthrough buffer be added. Possible values are "input", "output", or "both". By default, the buffer is added on the output side. This option is honored only for physical feedthroughs, not for logical feedthroughs.

-user_buffer *cell_ref*

Specifies the buffer library cell to use for all feedthroughs.

-nets *nets*

Specifies a collection of nets to work on. Only feedthroughs on ports on those nets will be isolated.

-logical

Inserts logical feedthroughs as well as physical feedthroughs. This option helps to eliminate assign statements in the output Verilog for all nonleaf hierarchical cells.

-verbose

Prints additional debugging information to the terminal.

DESCRIPTION

This command identifies all feedthroughs in the design and buffers them according to command options. By default, the command inserts buffers on a feedthrough net near the output port of the feedthrough.

EXAMPLES

The following example inserts buffers on all physical feedthroughs, with buffers on both input and output ports, with user buffer type "libcell1"

prompt> **add_feedthrough_buffers -buffer_side "both" -user_buffer "libcell1"**

The following example inserts buffers on all physical and logical feedthroughs.

prompt> **add_feedthrough_buffers -logical**

SEE ALSO

remove_feedthroughs(2)

add_group_repeaters

On route repeater planning for a group of nets or bundles.

SYNTAX

collection **add_group_repeaters**
-nets ***collection_of_nets*** | **-bundles** ***collection_of_bundles***
-repeater_distance ***length*** **[-first_distance** ***length*****]** **[-min_distance_repeater_to_load** ***length*****]** **[-tolerance** ***length*****]**
| **-relative_distance** ***distance_list***
| **-number_of_repeater_groups** ***number***
| **-location** ***location_list***
| **-cutlines** ***cutline_list***
[-allow_insertion_over_block]
-lib_cell ***repeater_lib_cell***
[-lib_cell_input ***lib_cell_pin*****]**
[-lib_cell_output ***lib_cell_pin*****]**
[-net_prefix ***prefix*****]**
[-cell_prefix ***prefix*****]**
[-horizontal_repeater_spacing ***spacing*****]**
[-vertical_repeater_spacing ***spacing*****]**
[-snap_pattern ***pattern*****]**
[-preview]
[-layer_cutting_spacing ***list*****]**
[-layer_cutting_distance ***list*****]**
[-honor_special_cell ***collection_of_lib_cells*****]**
[-smart_net_grouping | **-honor_user_net_grouping_id** ***list_of_group_ids*****]**

Data Types

collection_of_bundles	collection
collection_of_nets	collection
length	float
distance_list	list
location_list	list
cutline_list	list
number	integer
lib_cell_pin	string
prefix	string
spacing	integer
pattern	string
layer_cutting_spacing	list
layer_cutting_distance	list
honor_special_cell	collection
list_of_group_ids	list

ARGUMENTS

-nets *collection_of_nets*

Sepcifies multiple nets for adding group repeaters. This option is required.

-bundles *collection_of_bundles*

Sepcifies the multiple bundles for adding group repeaters. This option is required.

-repeater_distance *length* | -relative_distance *distance_list* | -location *location_list*

These three options are mutually exclusive. One of them must be used.

-repeater_distance *length*

Specifies the distance between repeater groups.

-first_distance *length*

Specifies the distance from drivers to the first group of repeaters. Use with -repeater_distance *length*.

-min_distance_repeater_to_load *length*

Specifies the minimum distance from last repeater group to the loads. This is only honored in two-pin nets. Use with -repeater_distance *length*.

-tolerance *length*

Specifies maximum delta distance over repeater distance. Use with -repeater_distance *length*.

-relative_distance *distance_list*

Specifies the distances from drivers to the first group of repeaters, then first group to second group of repeaters, and so on.

-number_of_repeater_groups *number*

Specifies the total number of repeater groups to be inserted. If the lib cell is inverting, the specified number must be even. This option can only be used in two-pin nets.

-location *location_list*

Specifies the center point coordinates for the locations of repeater group centers. The format is:

{*x1 y1 x2 y2 x3 y3* ...}

You must specify one location per repeater group when inserting buffers. You must specify two locations per inverter pair when inserting inverters. These locations can be any valid {x_coordinate y_coordinate} pair that is not far away from the placement region of the core area and is relative to the chip origin. These locations must be within a region from the route. Specify the coordinates in user units.

-cutlines *cutline_list*

Specifies line segments for the locations of repeater group centers. Cutline is a pair of points. It must be orthogonal to the routes. The format is:

{*{{x y1} {x y2}} {{x1 y} {x2 y}}* ...}

You must specify one cutline per repeater group when inserting buffers. You must specify two cutlines per inverter pair when

inserting inverters. These cutlines can be any valid pair of points {{x1 y1} {x2 y2}} that is orthogonal to the routes. Specify the coordinates in user units.

-allow_insertion_over_block

Specified that repeater groups can be inserted on top of blocks. This is optional.

-lib_cell ***lib_cell***

Specifies the library cell object to use as a repeater. The object is either a named library cell or a library cell collection.

The library cell must be either buffer or inverter. The command automatically determines the type of the library cell and inserts buffer or inverter pairs on routes based on the type.

-lib_cell_input ***lib_pin***

Specified the input pin name of the library cell. This is optional.

-lib_cell_output ***lib_pin***

Specified the output pin name of the library cell. This is optional.

-net_prefix ***prefix***

Specifies the prefix to be used for new nets that are created when buffers are inserted in the netlist. By default, the command uses "eco_net" as the prefix.

-cell_prefix ***prefix***

Specifies the prefix to be used for new buffers that are inserted in the netlist. By default, the command uses "eco_cell" as the prefix. This option cannot be used with the **-user_specified_buffers** option.

-horizontal_repeater_spacing ***spacing***

Specified the horizontal spacing between two adjacent repeaters in the horizontal direction in terms of number of site width. For example, when -snap_pattern is checkerboard, if the width of repeater is 9 cell site, then the minimum horizontal spacing is 9. If user specify it less than 9, it will automatically extend the spacing to 9. In order to make the checkerboard pattern, horizontal spacing should be 9,13,17,21... in this case.

-vertical_repeater_spacing ***spacing***

Specified the vertical spacing between repeaters in the vertical direction in terms of number of site height. When -snap_pattern is checkerboard, specified library cell is single site height, the minimum vertical spacing is 1 site height. In order to make a checkerboard pattern, veritcal spacing should be 1,5,9...in this case. When user specify vertical spacing as 4, it will automatically round to 5. For a double site height library cell, user can specify 2,6,10...to ensure checkerboard pattern.

-snap_pattern ***pattern***

Specified repeaters patterns within a group, support pattern: -checkerboard. If user don't specify -snap_pattern, then default snap pattern is checkerboard insertion pattern.

-preview

Display repeaters as gui annotation objects.

-layer_cutting_spacing

Specified the layer cutting spacing list, routes on output side of repeaters are right next to repeaters boundary, routes on input side of repeater can be further away from repeater boundary. Specify the option by using the following format:

{*{layer1 cutting_spacing_1} {layer2 cutting_spacing_2}* ...}

-layer_cutting_distance

Specified the layer cutting distance list, user can define input routes and output routes to cell center distances. Specify the option by using the following format:

{*{layer1 cutting_distance_in_1 cutting_distance_out_1} {layer2 cutting_distance_in_2 cutting_distance_out_2}* ...}

-honor_special_cell *lib_cell_collection*

Specified a collection of lib cells that repeaters will not inserted overlap with these cells.

-smart_net_grouping |-honor_net_group_id *list_of_group_ids*

These two options are mutually exclusive.

-smart_net_grouping

Use smart net grouping algorith to determine repeater locations. Must be used with -cutlines option.

-honor_user_net_grouping_id *list_of_group_ids*

Use existing group id on nets (net attribute eco_net_group_id). User specifies the descending priorities of net groups for repeater insertion.

DESCRIPTION

This command adds repeater groups for given nets or bundles. User specified distances between repeater groups or locations.

This command will creater the following collections: all newly-added repeaters (*agr_new_repeaters*), all the unrouted nets (*agr_unrouted_nets*) and all the nets that failed to insert buffers (*agr_unbuffered_nets*).

EXAMPLES

The following examples show three different usages of command options.

prompt> **add_group_repeaters -bundle {bundle_1 bundle2} -repeater_distance 100**

prompt> **add_group_repeaters -nets pr_nets -relative_distance {70 90 90 60}** \
-lib_cell lib/INV0

prompt> **add_group_repeaters -nets pr_nets -location {200 2500 300 2500 400 2500 500 2500}** \
-lib_cell lib/XOR4 -lib_cell_input "A4" -lib_cell_output "Z"

prompt> **add_group_repeaters -bundles [get_bundles] -lib_cell lib/BUF1 -number_of_repeater_groups 20**

prompt> **add_group_repeaters -nets pr_nets -lib_cell BUFFD4BWP -cutlines { {2632 1288} {2632 856} }**

prompt> **add_group_repeaters -nets pr_nets -lib_cell BUFFD4BWP -cutlines {{2632 1288} {2632 856}}** \
-smart_net_grouping

prompt> **add_group_repeaters -nets pr_nets -lib_cell BUFFD4BWP -cutlines {{2632 1288} {2632 856}}** \
-honor_user_net_grouping_id {3 2 1 6 5 4 0}

SEE ALSO

add_buffer_on_route(2)
create_group_repeaters_guidance(2)

add_macro_to_group

Add macro to macro group.

SYNTAX

status **add_macro_to_group**
-group macro_group
-macros list_of_macros

Data Types

list_of_macros collection of macros

ARGUMENTS

-group macro_group

Specifies macro group.

-macros list_of_macros

Specifies macros.

DESCRIPTION

This command add macros to the macro group. If the macro belongs to another macro group before the command, it will be removed from that macro group and added to the new macro group. If the macro group contains macros before the command, the added macros and existing macros must belong to the same placeable area, i.e, movebound, voltage area, and block. If a macro belongs to a macro array, the array will be added to the macro group.

EXAMPLES

The following command add macros M1, M2 to macro group groups.

prompt> **add_macro_to_groups** -group group1 -macros {M1 M2}

The following command add macro M1 to macro group group1. M1 belongs to a macro array array1. As the result, all macros of the array will be added to the macro group.

```
prompt> create_macro_array -num_rows 2 -num_cols 2 {M1 M2 M3 M4} \
  -name my_edit_group
prompt> add_macro_to_group -group group1 -macros M1
```

SEE ALSO

create_macro_groups(2)
remove_macros_from_group(2)

add_parameter

Define parameters that can be overridden by apply_power_model.

SYNTAX

string **add_parameter**
parameter_name
-default *default_value*
[-type *type_name*]
[-description *description text*]

Data Types

parameter_name	string
default_value	string
type_name	buildtime, runtime, or rate
description text	string

ARGUMENTS

parameter_name

Specifies the name of the parameter to be defined inside a power model.

-default *default_value*

Specifies the default value of the parameter if there is no parameter mapping specified in the **apply_power_model** command.

-type *type_name*

When -type option is specified, the parameter defined is used for system-level IP power model. The parameter scope is within the power model only and power models and power functions cannot access parameters that are defined outside of the power model in which they are used.

Three types of parameters 3 can be defined as follows:

1. Build time - for parameters that remain unchanged during run time
2. Run time - for parameters that can change during run time
3. Rate - for parameters that represent rate-based quantities that can change during run time

-description *text*

Text description of the parameter.

DESCRIPTION

The add_parameter command is used to define parameters for use within a power model. A parameter can be used just like a regular Tcl variable by prepending the parameter name with the $ keyword.

This command can only be defined within the **define_power_model** command and applied to a macro instance through the **apply_power_model** command. It is an error if this command is executed through the **load_upf** command.

The **-parameters** option of the **apply_power_model** command can change the value of a parameter during application of the power model on a macro instance. If the parameter is not found in the **-parameter** option, default value specifed in **add_parameter** will be used.

A power model provides a self-contained environment for variables. Parameters defined outside a power model are not visible inside the power model. Parameters defined within a power model are only visible inside the power model.

EXAMPLES

```
define_power_model MODEL -for {lib_cell} {
  add_parameter PD_NAME -default "PD" -description "power domain in model"
  add_parameter process -type buildtime -default 1.0
  add_parameter cache_miss -type rate -default 0.02
  create_supply_net VDD
  create_supply_net VSS
  create_supply_set SS -function {power VDD} -function {ground VSS}
  create_power_domain $PD_NAME -supply {primary SS} -include_scope
}

apply_power_model MODEL -parameters {{PD_NAME pd_macro}}
```

SEE ALSO

apply_power_model(2)
define_power_model(2)
report_power_model(2)

add_pins_to_virtual_connection

Connects pins or ports to the virtual connection.

SYNTAX

status **add_pins_to_virtual_connection**
-object *virtual_connection*
-pins *pin_port_list*

Data Types

virtual_connection collection
pin_port_list collection

ARGUMENTS

-object *virtual_connection*

Specifies one virtual connection.

-pins *pin_port_list*

Specifies a list of pins and ports to be connected to the virtual connection. The command errors out if any pin is connected to the existing virtual connection.

DESCRIPTION

This command connects pins to the specified virtual connection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples connect pins to the virtual connection.

prompt> **add_pins_to_virtual_connection** \

```
    -object [get_virtual_connections -name SNPS_vc_0] \
    -pins {lh1/ff3/Y U2/A U3/A MemData[0]}

prompt> add_pins_to_virtual_connection \
    -object [get_virtual_connections -name SNPS_vc_0] \
    -pins [get_pins {U2/A U3/A}]

prompt> add_pins_to_virtual_connection \
    -object [get_virtual_connections -name SNPS_vc_0] \
    -pins [get_ports MemData[0]]
```

SEE ALSO

create_virtual_connection(2)
get_attribute(2)
get_virtual_connections(2)
place_eco_cells(2)
remove_pins_from_virtual_connection(2)
remove_virtual_connections(2)
set_attribute(2)

add_port_protection_diodes

Adds diodes around the specified ports to protect them. One diode is added to each specified port.

SYNTAX

status **add_port_protection_diodes**
[-prefix *name*]
[-ignore_dont_touch]
-diode_lib_cell *lib_cell*
-port *ports*

Data Types

name string
lib_cell collection
ports collection

ARGUMENTS

-prefix *name*

Specifies the prefix used to identify the newly added diode cells. By default, no prefix is used and the tool assigns a cell name.

-ignore_dont_touch

Specifies that port protection diodes can be added to nets with the dont_touch attribute.

By default, port protection diodes are not inserted on nets that have the dont_touch attribute.

-diode_lib_cell *lib_cell*

Specifies the name of the library reference cell used for the new diodes. You can specify only a single library cell.

This is a required option.

-port *ports*

Specifies the ports to which to connect protection diodes.

This is a required option.

DESCRIPTION

The **add_port_protection_diodes** command performs the following tasks:

1. Creates the new diodes and adds them into the netlist.
2. Connects the new diodes to the specified ports.
3. Legalizes the diodes.
4. Marks the diodes as fixed.

Use this command in the preroute stage to prevent antenna violations.

Note that if this command succeeds, all of the inserted diodes are located at legal locations and marked as fixed. However, this does not necessarily mean that other movable standard cells are in legal locations. It is possible that some movable standard cells overlap with the inserted diodes. To legalize these standard cells, run the **place_opt**, **clock_opt** command or the **legalize_placement** command.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following example runs the **add_port_protection_diodes** command.

```
prompt> add_port_protection_diodes -prefix MY_DIODE \
  -diode_lib_cell [get_lib_cell XXX] -port [all_inputs]
```

SEE ALSO

report_port_protection_diodes(2)
remove_cells(2)
legalize_placement(2)

add_port_state

Defines the names and voltages of the possible states of a supply port for a UPF power state table.

SYNTAX

string **add_port_state**
supply_port_name
-state {*state_name state_value*}

Data Types

supply_port_name string
state_name string
state_value string

ARGUMENTS

supply_port_name

Specifies the name of the supply port. Hierarchical names are allowed. Use only supply port names previously listed in the **create_pst** command.

-state {*state_name state_value*}

Specifies the name and value of a state of the supply port. You can repeat this option in the command to define multiple supply states.

The supply state value can be specified as one of the following:

- A single floating-point number that represents the nominal voltage for the named state:

  -state {sLO 0.88}

- Three floating-point numbers that represent the minimum, nominal, and maximum voltages of the named state:

  -state {sLO 0.88 0.90 0.92}

- The keyword **off**, which represents the inactive state:

  -state {PDOWN off}

DESCRIPTION

The **add_port_state** command adds state information to a supply port for a UPF power state table. The command specifies the name of the supply port and the possible states of the port. Each state is specified as a state name and the voltage level for that state.

State name must meet UPF naming convention. Otherwise tool will issue an error.

You can specify the voltage level as a single nominal value, a set of three values (minimum, nominal, and maximum), or the keyword **off** to indicate the off state. The first state specified is the default state.

A port might be in any of the defined states and can take any voltage within a defined range, so a tool cannot determine the voltage to use from the power state table alone. In most tools, use the **set_voltage** command (a non-UPF command) to specify the operating voltage.

Use the **create_pst** command first, then the **add_port_state** command to specify the names and voltages of the possible states for each supply port, and finally the **add_pst_state** command to specify the allowed combinations of supply states. Here is an example:

```
# Create power state table, specify supply ports for table
create_pst MyPST -supplies { PN1 PN2 SOC/PN3 }

# Specify supply states and corresponding voltages
add_port_state PN1     -state {sLO 0.88}
add_port_state PN2     -state {sLO 0.88} -state {sHI 0.99}
add_port_state SOC/PN3 -state {sLO 0.88} -state {PDOWN off}

# Specify power states (allowed combinations of supply states)
add_pst_state S1 -pst MyPST -state {sLO sLO sLO}
add_pst_state S2 -pst MyPST -state {sLO sLO PDOWN}
add_pst_state S3 -pst MyPST -state {sLO sHI PDOWN}
```

The power state table defines the legal combinations of states that can exist at any given time during the operation of the design. The supply states sLO, sHI, and PDOWN represent the low voltage, high voltage, and power-down states of the supplies. The power states S1, S2, and S3 each refer to a specific combination of supply states for the supplies listed in the **create_pst** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the states and voltages of a supply port named VN1:

```
prompt> add_port_state VN1 \
     -state {active_state 0.88 0.90 0.92} \
     -state {off_state off}
```

SEE ALSO

add_pst_state(2)
create_pst(2)
report_pst(2)

add_power_state

Adds power state information to a supply set.

SYNTAX

status **add_power_state**
[-supply]
[-group]
[-domain]
object_name
-state *state_name*
{[-supply_expr {*supply_expression*}]
[-logic_expr {*logic_expression*}]
[-simstate {*simstate*}]
[-illegal]}
[-update]

Data Types

object_name string
state_name string
supply_expression string
logic_expression string
simstate string

ARGUMENTS

-supply | **-group** | **-domain** ***object_name***

Specifies the name of the supply set(-supply) or group(-group) or domain (-domain). The name should be a simple, non-hierarchical name.

-state ***state_name***

Specifies the name of the state of the supply set. The name should be a simple, non-hierarchical name.

-supply_expr {*supply_expression*}

Specifies the net and netstate. The **supply_expression** must use the following syntax if without -supply option

net == *netstate*

The **supply_expression** can use the following syntax if with -supply option

net == *netstate* && *net* == *netstate*

The *net* can be power or ground, and the *netstate* syntax must be one of the following:

status in V2.0 syntax: without -supply option and single == state
`{*status*}
`{*status*, *nom*}
`{*status*, *min*, *max*}
`{*status*, *min*, *nom*, *max*}

status in V3.0 syntax: with -supply option or multiple == state with && supported
{*status*}
{*status nom*}
{*status min max*}
{*status min nom max*}

- *status* can be OFF or FULL_ON
- *min*, *nom*, and *max* are floating-point numbers representing the minimum, nominal, and maximum voltages of the specified state, respectively.
- If the *status* is FULL_ON, at least one voltage must be defined. You can specify one or three voltage values. If three values are specified, they should be in increasing order.

-logic_expr {*logic_expression*} without -group/-domain option

Specifies a SystemVerilog Boolean expression in terms of logic nets and supply nets. The **-logic_expr** option does not affect implementation and is intended to be used by simulation tools, to read and write transparently.

-logic_expr {*logic_expression*} with -group/-domain option

Specifies expression in terms of group/domain state or supply set state. The **logic_expression** can use the following syntax with for group/domain state definition

supplySet/group/PST/domain == state && supplySet/group/PST/domain == state

-simstate {*simstate*}

Specifies a simstate for the power states associated with a supply set, valid values are NORMAL, CORRUPT_ON_ACTIVITY, CORRUPT, NOT_NORMAL, CORRUPT_STATE_ON_CHANGE, CORRUPT_STATE_ON_ACTIVITY, and CORRUPT_ON_CHANGE. The **-simstate** option does not affect implementation and is intended to be used by simulation, so the tool reads and writes the information specified with this option transparently.

-illegal

If specified, the power state is defined as illegal. If not specified (default) the state is legal state.

-update

Indicates this command provides additional information for a previous command with the same object_name and executed in the same scope.

DESCRIPTION

This command defines one or more power states of an object. Power states may be defined for a supply set, a power domain, a group.

If the *object_name* does not already exist as supply set or power state group or power domain, this command issues an error.

For supply set state, -update is supported for new state definition but it's optional.

e.g. prompt > **add_power_state SS1 -state SS1_state1 ...**

To add another SS1_state2, there are three ways to do it:

prompt > **add_power_state SS1 -state SS1_state1 ... -state SS1_state2 ...**
prompt > **add_power_state SS1 -state SS1_state2 ...**
prompt > **add_power_state SS1 -state SS1_state2 ... -update**

For group state, -update is mandatory for new state definition in this release.

prompt > **add_power_state -group GroupA -state A_state1 ...**

To add another A_state2, there are two ways to do it:

prompt > **add_power_state -group GroupA -state A_state1 ... -state A_state2 ...**
prompt > **add_power_state -group GroupA -state SS1_state2 ... -update**

Following without -update is an Error :

prompt > **add_power_state -group GroupA -state SS1_state2 ...**

For supply set state, -update is supported for sub-state conjunctions :

e.g. prompt > **add_power_state SS1 \ -state HPg {-supply_expr {(power == {OFF}) && (ground =={FULL_ON 0})}}**

Or use -update :

prompt > **add_power_state SS1** \
-state HPg {-supply_expr {(power == {OFF})}
prompt > add_power_state SS1 \
-state HPg {-supply_expr {(ground =={FULL_ON 0})}} -update

Following without -update will cause an Error for duplicated definition of state HPg

prompt > **add_power_state SS1** \
-state HPg {-supply_expr {(ground =={FULL_ON 0})}}

For group state, -update is also supported for sub-state conjunctions:

e.g. prompt> **add_power_state -group group_A \ -state lo2 {-logic_expr {SS1 == HPg && SS2 == HVp}**

Or use -update :

prompt > **add_power_state group1** \
-state lo2 {-logic_expr {SS1 == Hpg}}
prompt > **add_power_state group1** \
-state lo2 {-logic_expr {SS2 == HVp}} -update

We even can create an empty power state first, then update it later:

prompt>**add_power_state SS1 -state ON {}** \
prompt>**add_power_state SS1 ON {-supply_expr {power == FULL_ON}} -update**\

The domain state has same usage as group/supply set, with -update option.

prompt > **add_power_state domain1** \
-state hi_state {-logic_expr {SS1 == HPg && primary == HVp}}

```
prompt > add_power_state domain1 \
      -state off_state {-logic_expr {SS1 == HPg && primary == OFF}} -update
prompt > add_power_state group1 -state hi {-logic_expr {domain1 == hi_state}}
```

EXAMPLES

The following example shows the states and voltages of a supply set, SS1:

```
prompt> add_power_state SS1 \
      -state HVp {-supply_expr {power == `{FULL_ON, 0.88, 0.90, 0.92}}}

prompt> add_power_state SS1 \
      -state HPg {-supply_expr {ground == `{OFF}}}

prompt> add_power_state SS1 -supply \
      -state HPg {-supply_expr {(power == {OFF}) && (ground =={FULL_ON 0})}}

prompt> add_power_state -group group_A \
      -state lo2 {-logic_expr {SS1 == HPg && SS1 == HVp}
```

SEE ALSO

add_pst_state(2)
create_pst(2)
create_supply_set(2)
report_pst(2)

add_pst_state

Defines the states of each of the supply ports or nets for one possible state of the design for a UPF power state table.

SYNTAX

status **add_pst_state**
state_name
-pst *table_name*
-state *supply_states*

Data Types

state_name	string
table_name	string
supply_states	list

ARGUMENTS

state_name

Specifies a name for the power state being defined in the command.

-pst *table_name*

Specifies the name of the power state table previously defined by the **create_pst** command. The power state being defined is added to this power state table.

-state *supply_states*

Lists the supply state names in the order corresponding to the **-supplies** option listing in the **create_pst** command. For the power state being defined, the listed supply states are active for the respective supplies. The supply state names in the list must be defined by previous **add_port_state** commands.

DESCRIPTION

The **add_pst_state** command defines the states of each of the supply ports or supply nets for one possible power state configuration of the design for a UPF power state table.

The command specifies a name for the power state, the name of the power state table previously created by the **create_pst** command, and the states of the supply ports in the same order listed in the **create_pst** command. The voltages corresponding to these supply states are defined by the **add_pst_state** command.

The following script demonstrates the power state definition flow:

```
# Create power state table, specify supply ports for table
create_pst MyPST -supplies { PN1 PN2 SOC/PN3 }

# Specify supply states and corresponding voltages
add_port_state PN1     -state {sLO 0.88}
add_port_state PN2     -state {sLO 0.88} -state {sHI 0.99}
add_port_state SOC/PN3 -state {sLO 0.88} -state {PDOWN off}

# Specify power states (allowed combinations of supply states)
add_pst_state S1 -pst MyPST -state {sLO sLO sLO}
add_pst_state S2 -pst MyPST -state {sLO sLO PDOWN}
add_pst_state S3 -pst MyPST -state {sLO sHI PDOWN}
```

The power state table defines the legal combinations of states that can exist at any given time during the operation of the design. The supply states sLO, sHI, and PDOWN represent the low voltage, high voltage, and power-down states of the supplies. The power states S1, S2, and S3 each refer to a specific combination of supply states for the supplies listed in the **create_pst** command.

For a group of supply ports and supply nets directly connected together, the allowable supply states are derived from the shared pool of supply states commonly owned by the members of the group.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following script specifies four allowed combinations of supply states for the power state table named MyPST:

```
add_pst_state S1 -pst MyPST -state {sLO sLO sLO}
add_pst_state S2 -pst MyPST -state {sLO sLO PDOWN}
add_pst_state S3 -pst MyPST -state {sLO sHI PDOWN}
add_pst_state S4 -pst MyPST -state {sHI sLO PDOWN}
```

SEE ALSO

add_port_state(2)
create_pst(2)
report_pst(2)

add_redundant_vias

Replaces single-cut vias with multiple-cut via arrays. It can also replace a single-cut via with another single-cut via that has a different contact code.

SYNTAX

status **add_redundant_vias**
[-effort low | medium | high]
[-list_only true | false]
[-timing_preserve_nets *collection_of_nets*]
[-timing_preserve_setup_slack_threshold *slack_value*]
[-timing_preserve_hold_slack_threshold *slack_value*]
[-nets *collection_of_nets*]

Data Types

collection_of_nets	collection
slack_value	float

ARGUMENTS

-effort low | **medium** | **high**

Specifies the effort used to perform redundant via insertion.

The default is medium.

-list_only true | **false**

Controls whether the tool performs redundant via insertion or lists the default via mappings to be used for redundant via insertion.

By default (**false**), the tool performs redundant via insertion.

-timing_preserve_nets *collection_of_nets*

Specifies the nets to preserve timing on.

If you specify this option, the command considers the specified nets as timing critical and skips redundant via insertion on them.

-timing_preserve_setup_slack_threshold *slack_value*

Specifies the setup slack threshold for timing critical nets. The unit of the threshold is in library units.

If you specify this option, the command skips redundant via insertion on all nets with a worst setup slack less than or equal to this value.

-timing_preserve_hold_slack_threshold *slack_value*

Specifies the hold slack threshold for timing critical nets. The unit of the threshold is in library units.

If you specify this option, the command skips redundant via insertion on all nets with a worst hold slack less than or equal to this value.

-nets ***collection_of_nets***

Specifies the nets on which to perform redundant via insertion.

By default, redundant via insertion is performed on all nets.

DESCRIPTION

This command replaces one set of vias in the design with another set. By default, all single-cut vias are replaced with 2-cut via arrays. You can change the behavior by using the **add_via_mapping** command to specify the exact via mapping to be used by this command. From a netlist connectivity point of view, using a single-cut via is no different from using a multiple-cut via array, so additional via cuts in a multiple-cut via array are regarded as redundant vias from this point of view. However, redundant vias can increase yield during manufacturing and are an important design-for-manufacturing (DFM) feature.

The command tries to replace vias when the new via or via array does not introduce any design rule violations. However, sometimes new violations are introduced by accident and the command cleans them up later by performing search and repair.

EXAMPLES

The following example replaces all single cut vias by 2-cut via arrays.

```
prompt> add_redundant_vias
```

The following example replaces single-cut vias named VIA23 with 1x3 via arrays named VIA23 and single-cut vias named VIA34 with 5x1 via arrays named VIA34.

```
prompt> add_via_mapping \
  -from {VIA23 1x1} -to {VIA23 1x3}
prompt> add_via_mapping \
  -from {VIA34 1x1} -to {VIA34 5x1}
prompt> add_redundant_vias
```

The following example replaces single-cut vias named VIA23 with 1x2 via arrays named VIA23 on the net named net1.

```
prompt> add_via_mapping \
  -from {VIA23 1x1} -to {VIA23 1x2}
prompt> add_redundant_vias -nets net1
```

SEE ALSO

add_via_mapping(2)

add_shield_association

Associates shielding shapes and vias to shielded nets.

SYNTAX

status **add_shield_association**
-objects *shape_and_via_list*
-nets *net_list*

Data Types

shape_and_via_list collection
net_list collection

ARGUMENTS

-objects *shape_and_via_list*

Specifies a list of shielding shapes and vias to add to shield association.

-nets *net_list*

Specifies a list of shielded nets to add to shield association.

DESCRIPTION

This command associates the specified shielding shapes and vias to the shielded nets.

EXAMPLES

The following example adds shielding shape 'shape1' and shielded net 'net1' to shield association.

prompt> **add_shield_association -objects shape1 -nets net1**
1

The following example adds shielding via 'via1' and shielded net 'net1' to shield association.

prompt> **add_shield_association -objects via1 -nets net1**

1

SEE ALSO

remove_shield_association(2)

add_spare_cells

Inserts spare cells into the current design.

SYNTAX

status **add_spare_cells**
-cell_name *name*
-lib_cell *reference*
[-num_instances *number*]
-num_cells {*reference number reference number* ...}
[-hier_cell *hierarchy_name*]
[-boundary *{ {{llx lly} {urx ury}} | {{x1 y1} {x2 y2} ...} }*]
[-voltage_areas *voltage_area_list*]
[-ignore_blockage_types *blockage_type_list*]
[-density_aware_ratio *percentage*]
[-input_pin_connect_type *tie_low* | *tie_high* | *open*]
[-repetitive_window {width height}]
[-random_distribution]

Data Types

name string
reference collection
number integer
hierarchy_name collection
llx float
lly float
urx float
ury float
x1 float
y1 float
x2 float
y2 float
voltage_area_list list
blockage_type_list list
percentage integer

ARGUMENTS

-cell_name *name*

Specifies the name prefix for the spare cells. An integer suffix is automatically appended to the name prefix for each instance. When a name is generated for a spare cell and the name already exists, the tool automatically assigns a unique name for the spare cell.

-lib_cell ***reference***

Specifies the library cells to insert as spare cells. The input can be names or collections. Use this option together with the **-num_instances** option. This option is mutually exclusive with the **-num_cells** option.

-num_instances ***number***

Specifies the number of instances of each library cell to add as spare cells. Use this option together with the **-lib_cell** option. This option is mutually exclusive with the **-num_cells** option.

-num_cells {*reference number reference number ...*}

Specifies library cell names and the number of instances to insert for each unique library cell. The valid format is {*ref1 num1 ref2 num2 ...*}. This option is mutually exclusive with the **-lib_cell** and **-num_instance** options.

-hier_cell ***hierarchy_name***

Specifies the hierarchy in which to insert the spares cells. The **-hier_cell**, **-boundary** and **-voltage_areas** options are mutually exclusive.

-boundary ***{ {{llx lly} {urx ury}} | {{x1 y1} {x2 y2} ...} }***

Specifies the boundary in which to insert the spares cells. Only one rectangle or rectilinear polygon is allowed for this option. When specifying this option, new spare cells are inserted inside voltage areas within the specified boundary. The **-hier_cell**, **-boundary** and **-voltage_areas** options are mutually exclusive.

-voltage_areas ***voltage_area_list***

Specifies the voltage areas in which the spares cells are added. The new spare cells are added to these specified voltage areas. The **-hier_cell**, **-boundary** and **-voltage_areas** options are mutually exclusive.

-ignore_blockage_types ***blockage_type_list***

Specifies a list of blockage types. The command allows spare cells to be added inside the blockage if blockage type is specified by the option. The command supports all blockage types same as create_placement_blockage, but does not honor any placement constraint that the blockage has. If the option is not used, the spare cells can not be added inside any placement blockage.

-density_aware_ratio ***percentage***

Specifies the percentage of spare cells to be added with density aware. This option allows user to control the percentage of density aware for adding spare cells. The percentage means the percentage of spare cells will be added considering density of existing cells, and the rest will be added evenly. The two spreading ways are both blockage aware.

- The default percentage is 100, that means all new spare cells are added with considering cell density;
- 20 means 20% spare cells will be added based on cell density and 80% spare cells will be added evenly across the design;
- 0 means all new spare cells are added evenly without considering cell density.

-input_pin_connect_type ***tie_low*** **|** ***tie_high*** **|** ***open***

Specifies the connection type of input pins of new spare cells. The valid values are **tie_low**, **tie_high** and **open**.

- **tie_low**: input pins are connected to tie low net.
- **tie_high**: input pins are connected to tie high net.
- **open**: input pins are kept open.

If this option is not specified, the default value is **tie_low**.

-repetitive_window *{width height}*

Add and place spare cells through repetitive windows. The *width* and *height* are distance with micron unit. When this option is specified, spare cells with the specified number and types will be added to each window. And the window will apply repetitively start from the low-left of the placement area. The option *-lib_cell* with *-num_instances* or option *-num_cells* can be used to specify the number and types of spare cells to be added within each window. The specified number is considered the number in each window. For windows that cut by blockage or other voltage areas and windows

For windows whose placeable area is not the same as window size, the cell number will be calculated on the area ratio.

- Windows that cut by blockages or other voltage areas.
- Windows that near the boundary edge when the boundary size is not integral multiple of window size.

-random_distribution

Place all newly added spare cells randomly. The *-random_distribution* and *-density_aware_ratio* options are mutually exclusive. When the option is used, the tool will ignore the current cell density and spare cells are placed with random distribution.

DESCRIPTION

This command adds a specified number of instances of the library cell as spare cells in a design. It adds the spare cells to the logical netlist and places them in corresponding physical regions as evenly as possible, and avoids all placement blockages.

Specifying more than one library cell allows you to place the instances of different library cells evenly throughout the regions. The library cell must exist in your link libraries.

The **-lib_cell** and **-num_instances** options are used together to insert spare cells with same number of instances for different library cells. To add spare cells with a different number of instances from different library cells, use the **-num_cells** option. You must use one of the **-lib_cell** and **-num_cells** options to specify library cells. The **-num_cells** option is mutually exclusive with the **-lib_cell** and **-num_instances** options.

This command provides three options to distribute the spare cells in different ways:

- **-hier_cell** : Spare cells are placed around the existing placed cells of the specified hierarchy. And these spare cells are also placed inside the regions of the voltage area of the specified hierarchy.
- **-boundary** : Spare cells are added to the voltage areas evenly within the specified boundary. These new cells are also added to the corresponding hierarchies.
- **-voltage_areas** : Spare cells are added to the specified voltage areas evenly. These new cells are also added to corresponding hierarchies.
- If none of these three options is specified, the default behavior is to distribute new spare cells to all voltage areas including the default voltage area for the design.

The **-hier_cell**, **-boundary** and **-voltage_areas** options are mutually exclusive.

The new spare cells will not be legalized. You can use **place_eco_cells -legalize_only** or **legalize_placement** to legalize these cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example adds 20 instances of a library cell *and2* with the prefix *SpareCell* to the current design.

```
prompt> add_spare_cells -cell_name SpareCell -lib_cell and2 -num_instances 20
```

The following example adds 15 instances of library cells *and2* and *nor2* with the prefix *SpareCell* in the hierarchical cell *I_SUB*.

```
prompt> add_spare_cells -cell_name SpareCell \
  -lib_cell {and2 nor2} -num_instances 15 -hier_cell I_SUB
```

The following example inserts 10 instances of library cell *and1* and 5 instances of library cell *nor1* with the prefix *SpareCell* in the boundary.

```
prompt> add_spare_cells -num_cells {and1 10 nor1 5} \
  -cell_name SpareCell  -boundary {{5.04 2.52} {120.06 80.4}}
```

The following example inserts 10 instances of library cell *and1* and 5 instances of library cell *nor1* with the prefix *SpareCell* in the specified voltage areas.

```
prompt> add_spare_cells -num_cells {and1 10 nor1 5} \
  -cell_name SpareCell -voltage_areas pd2
```

The following example uses the **-ignore_blockage_types** option to allow spare cells to be added inside the blockage with user-specified type.

```
prompt> add_spare_cells -num_cells {and1 10 nor1 5} \
  -cell_name SpareCell -ignore_blockage_types {soft partial}
```

The following example uses the **-density_aware_ratio** option to allow user to control the percentage of spare cells to be added with density aware.

```
prompt> add_spare_cells -num_cells {and1 10 nor1 5} \
  -cell_name SpareCell -voltage_areas pd2 -density_aware_ratio 100
prompt> add_spare_cells -num_cells {and1 10 nor1 5} \
  -cell_name SpareCell -voltage_areas pd2 -density_aware_ratio 80
```

The following example uses the **-input_pin_connect_type** option to control the connection type of the input pins of the new spare cells.

```
prompt> add_spare_cells -num_cells {and1 10 nor1 5} \
  -cell_name SpareCell -input_pin_connect_type tie_high
```

The following example uses the **-repetitive_window** option to add spare cells by repetitive window.

```
prompt> add_spare_cells -num_cells {and1 10 nor1 5} \
  -cell_name SpareCell -repetitive_window {20 20}
```

The following example uses the **-random_distribution** option.

```
prompt> add_spare_cells -num_cells {and1 10 nor1 5} \
  -cell_name SpareCell -random_distribution
```

SEE ALSO

create_placement_blockage(2)
legalize_placement(2)
place_eco_cells(2)
spread_spare_cells(2)

add_state_transition

Defined named state transitions between power states of an object.

SYNTAX

status **add_state_transition**
[-supply]
[-domain]
[-group]
object_name
-transition {*transition_name*
[-from *from_list* -to *to_list*]
[-paired {*from_state to_state*}]
[-through *through_list*]
[-legal]
[-illegal]}
[-update]

Data Types

object_name string
transition_name string
from_list list
to_list list
through_list list
from_state string
to_state string

ARGUMENTS

-supply | **-domain** | **-group** ***object_name***

Specifies the name of the supply set(-supply) or domain(-domain) or group (-group). If none of the type is specified, it could be any UPF object which contains the power state. The name should be a simple, non-hierarchical name.

transition_name

Specifies the simple name of transition.

-from ***from_list***

Specifies an unordered list of power state names active before a state transition. Empty list expands to all named power states for the specified object.

-to *to_list*

Specifies an unordered list of power state names active after a state transition. Empty list expands to all named power states for the specified object.

-paired {*from_state to_state*}

Specifies a list of from-state name and to-state name pairs.

-through *through_list*

Specifies list of intermediate states. This option is only applicable when -from/-to options are specified. The effective transition sequence is -from, -through, -to.

-legal | -illegal

Specify the legality of the transition being defined as either legal or illegal. The default is -legal.

-update

Indicates this command provides additional information for a previous command withthe same object_name and executed in the same scope.

DESCRIPTION

The add_state_transition command defines named state transitions between power states of an object.

This is a new command added in UPF 3.0, it replaces the describe_state_transition.

EXAMPLES

The following example shows how to add state transitions:

```
prompt> add_state_transition -domain PDA \
 -transition {turn_on -from OFF_MODE -to NORMAL_MODE}
 -transition {suspend -from NORMAL_MODE -to SLEEP_MODE}
 -transition {resume -from SLEEP_MODE -to NORMAL_MODE}
 -transition {turn_off -from NORMAL_MODE -to OFF_MODE}
prompt> add_state_transition -domain PDA -update\
 -transition {error1 -from OFF_MODE -to SLEEP_MODE -illegal}
```

SEE ALSO

add_power_state(2)
describe_state_transition(2)

add_supply_state

Defines the names and voltages of the possible states of a supply object for a UPF power state table.

SYNTAX

string **add_supply_state**
supply_name
-state {*state_name state_value*}

Data Types

supply_name string
state_name string
state_value string

ARGUMENTS

supply_name

Specifies the name of the supply port, supply net or supply set function. Hierarchical names are allowed.

-state {*state_name state_value*}

Specifies the name and value of a state of the supply. You can repeat this option in the command to define multiple supply states.

The supply state value can be specified as one of the following:

- A single floating-point number that represents the nominal voltage for the named state:

  -state {sLO 0.88}

- The keyword **off**, which represents the inactive state:

  -state {PDOWN off}

DESCRIPTION

The **add_supply_state** command adds state information to a supply object for a UPF power state table. The command specifies the name of the supply object and the possible states of the object. Each state is specified as a state name and the voltage level for that state.

You can specify the voltage level as a single nominal value, or the keyword **off** to indicate the off state. The first state specified is the

default state.

A supply object might be in any of the defined states and can take any voltage within a defined range, so a tool cannot determine the voltage to use from the power state table alone. In most tools, use the **set_voltage** command (a non-UPF command) to specify the operating voltage.

Use the **create_pst** command first, then the **add_supply_state** command to specify the names and voltages of the possible states for each supply, and finally the **add_pst_state** command to specify the allowed combinations of supply states. Here is an example:

```
# Create power state table, specify supply ports or nets for table
create_pst MyPST -supplies { PN1 PN2 SOC/PN3 }

# Specify supply states and corresponding voltages
add_supply_state PN1     -state {sLO 0.88}
add_supply_state PN2     -state {sLO 0.88} -state {sHI 0.99}
add_supply_state SOC/PN3 -state {sLO 0.88} -state {PDOWN off}

# Specify power states (allowed combinations of supply states)
add_pst_state S1 -pst MyPST -state {sLO sLO sLO}
add_pst_state S2 -pst MyPST -state {sLO sLO PDOWN}
add_pst_state S3 -pst MyPST -state {sLO sHI PDOWN}
```

The power state table defines the legal combinations of states that can exist at any given time during the operation of the design. The supply states sLO, sHI, and PDOWN represent the low voltage, high voltage, and power-down states of the supplies. The power states S1, S2, and S3 each refer to a specific combination of supply states for the supplies listed in the **create_pst** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the states and voltages of a supply set function named ss1.power:

```
prompt> add_supply_state ss1.power \
    -state {active_state1 0.88} \
    -state {active_state2 0.92} \
    -state {off_state off}
```

SEE ALSO

add_port_state(2)
add_pst_state(2)
create_pst(2)
report_pst(2)

add_tie_cells

Inserts tie cells into the current design.

SYNTAX

status **add_tie_cells**
[-objects *object_name_or_collection*]
[-tie_high_lib_cells *lib_cells*]
[-tie_low_lib_cells *lib_cells*]

Data Types

object_name_or_collection collection
lib_cells list

ARGUMENTS

-objects *object_name_or_collection*

Specifies the objects for tie cell insertion. Currently supported object types are pin, port, cell and lib_cell. If this option is not used, all constant pins in current design will be considered for tie cell insertion. If the object type is cell, only the constant pins on the cell will be considered. If the object type is lib_cell, only the constant pins of the cells that match the library cells will be considered.

-tie_high_lib_cells *lib_cells*

Limits the tie high cells that can be used for this invocation of the command. When this argument is supplied, tie cell insertion will only use tie cells that are neither don't use nor don't touch, are for optimization purpose and in the library cell list. A library cell in the list that is not a tie cell will not be used.

-tie_low_lib_cells *lib_cells*

Limits the tie low cells that can be used for this invocation of the command. When this argument is supplied, tie cell insertion will only use tie cells that are neither don't use nor don't touch, are for optimization purpose and in the library cell list. A library cell in the list that is not a tie cell will not be used.

DESCRIPTION

This command adds tie cells to drive constant pins in the design, if there are useable tie cells in the library.

SEE ALSO

set_lib_cell_purpose(2)

add_to_bound

Assigns cells, ports, and groups to a bound in the current design.

SYNTAX

integer **add_to_bound**
bound_object
cell_and_port_and_group_list

Data Types

bound_object	list
cell_and_port_and_group_list	list

ARGUMENTS

bound_object

Specifies the name or collection of a single bound. The bound object can be specified by using the **get_bounds** command. However it must return a collection of one and only one bound object.

cell_and_port_and_group_list

Specifies a list of cells, ports, and groups to assign to the bound. The list may contain names, patterns, or collections. It is an error to include a port which has already been assigned to another bound. If the bound is a move bound, it is an error to include a cell which has already been explicitly assigned to another move bound. If this bound is a group bound, it is an error to include a cell which has already been explicitly assigned to another group bound. If a cell is hierarchical, its child cells are also implicitly in the bound, as well as any cells added to its hierarchy in the future, unless the child cell is assigned to another bound. Cells that belong to a relative placement group cannot be individually added to a bound. All cells of a relative placement group must be added to a bound together. Group objects are only valid when adding to a repelling bound. Furthermore, repelling bounds may contain either cells and ports or groups, but not a mix.

DESCRIPTION

The **add_to_bound** command assigns cells, ports, and groups to an existing bound.

You can change the floorplan by assigning cells to the bound. The placer honors the new bound cells during coarse placement.

EXAMPLES

The following example assigns cells to a bound named "movebound1".

```
prompt> add_to_bound movebound1 { inv1 add1 }
1
```

The following example assigns the hierarchical cell "mid" to the bound. The bound implicitly contains all cells within "mid".

```
prompt> add_to_bound movebound1 [get_cells mid]
1
```

The following example assigns all cells of a relative placement group "rp1" to the bound.

```
prompt> add_to_bound movebound1 [get_cells -of_objects rp1]
1
```

The following example assigns all repelling bound groups to the bound

```
prompt> add_to_bound repellingGroupBound1 [get_groups -filter "is_repelling_bound_group==true"]
```

SEE ALSO

create_bound(2)
get_bounds(2)
remove_bounds(2)
remove_from_bound(2)
report_bounds(2)

add_to_bundle

Adds objects to the specified bundle.

SYNTAX

status **add_to_bundle**
-bundle *bundle*
[-prepend | -before_object *before_object*]
objects

Data Types

bundle collection
before_object collection
objects collection

ARGUMENTS

-bundle *bundle*

Specifies the bundle to which to add the objects.

-prepend

Adds the objects as the first objects in the bundle. This option is mutually exclusive with the **-before_object** option. By default, objects are added to the end of the bundle.

-before_object *before_object*

Adds the objects before the specified object in the bundle. This option is mutually exclusive with the **-prepend** option. By default, objects are added to the end of the bundle.

objects

Specifies the objects to add to the bundle. The objects must be nets, bundles or supernets.

The objects are added to the bundle in the order in which they are specified.

DESCRIPTION

This command adds one or more objects to the specified bundle. By default, the objects are added at the end of the bundle. You can insert objects at the start of the bundle with the **-prepend** option, or insert the objects before a specified object in the bundle with the

-before_object option.

A different type of object than the type of bundle can't be added to the bundle.

EXAMPLES

The following example adds an object to the end of a bundle.

```
prompt> add_to_bundle -bundle [get_bundles BUNDLE_5] [get_nets Clk]
1
```

The following example adds an object to the start of a bundle.

```
prompt> add_to_bundle -bundle [get_bundles BUNDLE_5] \
  -prepend [get_nets Reset]
1
```

SEE ALSO

create_bundle(2)
get_bundles(2)
get_nets(2)
remove_bundles(2)
remove_from_bundle(2)
report_bundles(2)

add_to_collection

Adds objects to a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

collection **add_to_collection**
[-unique]
collection1
object_spec

Data Types

collection1	collection
object_spec	list

ARGUMENTS

-unique

Indicates that duplicate objects are to be removed from the resulting collection. By default, duplicate objects are not removed.

collection1

Specifies the base collection to which objects are to be added. This collection is copied to the result collection, and objects matching *object_spec* are added to the result collection. The *collection1* option can be the empty collection (empty string), subject to some constraints, as explained in the DESCRIPTION section.

object_spec

Specifies a list of named objects or collections to add.

If the base collection is heterogeneous, only collections can be added to it.

If the base collection is homogeneous, the object class of each element in this list must be the same as in the base collection. If it is not the same class, it is ignored. From heterogeneous collections in the *object_spec*, only objects of the same class of the base collection are added. If the name matches an existing collection, the collection is used. Otherwise, the objects are searched for in the database using the object class of the base collection.

The *object_spec* has some special rules when the base collection is empty, as explained in the DESCRIPTION section.

DESCRIPTION

The **add_to_collection** command allows you to add elements to a collection. The result is a new collection representing the objects in the *object_spec* added to the objects in the base collection.

Elements that exist in both the base collection and the *object_spec*, are duplicated in the resulting collection. Duplicates are not removed unless you use the *-unique* option. If the *object_spec* is empty, the result is a copy of the base collection.

If the base collection is homogeneous, the command searches in the database for any elements of the *object_spec* that are not collections, using the object class of the base collection. If the base collection is heterogeneous, all implicit elements of the *object_spec* are ignored.

When the *collection1* argument is the empty collection, some special rules apply to the *object_spec*. If the *object_spec* is non-empty, there must be at least one homogeneous collection somewhere in the *object_spec* list (its position in the list does not matter). The first homogeneous collection in the *object_spec* list becomes the base collection and sets the object class for the function. The examples show the different errors and warnings that can be generated.

The **append_to_collection** command has similar semantics as the **add_to_collection** command; however, the **append_to_collection** command can be much more efficient in some cases. For more information about the command, see the man page.

For background on collections and querying of objects, see the **collections** man page.

EXAMPLES

The following example from PrimeTime uses the **get_ports** command to get all of the ports beginning with 'mode' and then adds the "CLOCK" port.

```
pt_shell> set xports [get_ports mode*]
{"mode[0]", "mode[1]", "mode[2]"}
pt_shell> add_to_collection $xports [get_ports CLOCK]
{"mode[0]", "mode[1]", "mode[2]", "CLOCK"}
```

The following example from PrimeTime adds the cell u1 to a collection containing the SCANOUT port.

```
pt_shell> set so [get_ports SCANOUT]
{"SCANOUT"}
pt_shell> set u1 [get_cells u1]
{"u1"}
pt_shell> set het [add_to_collection $so $u1]
{"u1"}
pt_shell> query_objects -verbose $het
{"port:SCANOUT", "cell:u1"}
```

The following examples show how the **add_to_collection** command behaves when the base collection is empty. Adding two empty collections yields the empty collection. Adding an implicit list of only strings or heterogeneous collections to the empty collection generates an error message, because no homogeneous collections are present in the *object_spec* list. Finally, as long as one homogeneous collection is present in the *object_spec* list, the command succeeds, even though a warning message is generated. The example uses the variable settings from the previous example.

```
pt_shell> sizeof_collection [add_to_collection "" ""]
0

pt_shell> set A [add_to_collection "" [list a $het c]]
Error: At least one homogeneous collection required for argument 'object_spec'
       to add_to_collection when the 'collection' argument is empty (SEL-014)
```

```
pt_shell> add_to_collection "" [list a $het $sp]]
Warning: Ignored all implicit elements in argument 'object_spec'
        to add_to_collection because the class of the base collection
        could not be determined (SEL-015)
{"SCANOUT", "u1", "SCANOUT"}
```

SEE ALSO

append_to_collection(2)
collections(2)
query_objects(2)
remove_from_collection(2)
sizeof_collection(2)

add_to_edit_group

Adds objects to an edit group in the current design.

SYNTAX

collection **add_to_edit_group**
edit_group_object
object_list

Data Types

edit_group_object list
object_list list

ARGUMENTS

edit_group_object

Specifies the name or collection of a single edit group. The edit group object can be specified by using the **get_edit_groups** command, however it must return a collection of exactly one edit group object.

object_list

Specifies a list of objects to add to the edit group. The list can contain names, patterns, or collections. It is an error to include an object which is already contained within another edit group or one that belongs to a different design.

DESCRIPTION

The **add_to_edit_group** command adds objects to an existing edit group. The command returns a collection containing the modified edit group as the only object, or TCL_ERROR if the command fails.

EXAMPLES

The following example adds objects to an edit group named "edit_group1"

prompt> **add_to_edit_group edit_group1 { U31 voltage_group2 }**
{"edit_group1"}

SEE ALSO

create_edit_group(2)
get_edit_groups(2)
remove_edit_groups(2)
remove_from_edit_group(2)
report_edit_groups(2)

add_to_group

Adds objects to the group in the current design.

SYNTAX

integer **add_to_group**
group
object_list

Data Types

group list
object_list list

ARGUMENTS

group

Specifies the name or collection of groups. The group object can be specified by using the **get_groups** command.

object_list

Specifies a list of objects to add to the group. The list can contain names, patterns, or collections. It is an error to include an object that does not belong to the current design.

Supported object classes are cell, pin, port, net, module, shape, placement_blockage, site_row, via, track, bound, bound_shape, routing_rule, routing_blockage, routing_guide, io_guide, io_ring, voltage_area, voltage_area_shape, pin_blockage, pin_guide, pg_region, edit_group, terminal, routing_corridor, routing_corridor_shape, grid, matching_type, site_array, supernet, shaping_blockage, constraint_group, topological_constraint, via_region, topology_edge, topology_node, overlap_blockage and group.

Supported object classes for repelling bound groups are cell.

DESCRIPTION

The **add_to_group** command adds objects to the group. The command returns the number of objects added into the group. If the command fails, it returns a TCL_ERROR.

EXAMPLES

The following example adds two nets n12, n13 to the group named "group1"

```
prompt> add_to_group group1 [ get_nets { n12 n13} ] }
2
```

SEE ALSO

create_group(2)
get_groups(2)
remove_groups(2)
remove_from_group(2)
report_groups(2)

add_to_io_guide

Adds pads to an I/O guide in the current design.

SYNTAX

int **add_to_io_guide**
[-before *pad_object*]
io_guide_object
pad_cell_list

Data Types

pad_object pad cell
io_guide_object list
pad_cell_list list

ARGUMENTS

-before *pad_object*

Specifies the name or collection of a single library cell. The pads are inserted before the specified pad cell. By default, new cells are added at the end of the list.

io_guide_object

Specifies the name or collection of a single I/O guide. The *io_guide_object* can be specified by using the **get_io_guides** command. However it must return a collection of one and only one io_guide object.

pad_cell_list

Specifies a list of pads to add to the I/O guide. The list can contain names, patterns, or collections. It is an error to include a cell or port which has already been assigned to another io_guide or one that belongs to a different design.

DESCRIPTION

The **add_to_io_guide** command adds pad cells to an existing I/O guide. You can change the floorplan by adding cells to the I/O guide. The I/O placer honors the new io_guide cells during coarse placement.

The command returns 1 if the command succeeds, 0 if it fails, or TCL_ERROR if there is a command syntax error.

EXAMPLES

The following example adds cells to an io_guide named "io_guide1"

```
prompt> add_to_io_guide io_guide1 { bus_4 }
1
```

The following example adds cells to an io_guide named "io_guide1" before pad3.

```
prompt> add_to_io_guide -before bus_4 io_guide1 { bus_2 bus_3 }
1
```

SEE ALSO

create_io_guide(2)
get_io_guides(2)
remove_from_io_guide(2)
report_io_guides(2)
remove_io_guides(2)

add_to_io_ring

Adds guides to an I/O ring in the current design.

SYNTAX

int **add_to_io_ring**
io_ring_object
guide_list

Data Types

io_ring_object list
guide_list list

ARGUMENTS

io_ring_object

Specifies the name or collection of a single I/O ring. The *io_ring_object* can be specified by using the **get_io_rings** command. However it must return a collection of one and only one io_ring object.

guide_list

Specifies a list of I/O guides to add to the I/O ring. The list can contain I/O guide names, patterns, or collections. It is an error to include a I/O guide which has already been assigned to another I/O ring.

DESCRIPTION

The **add_to_io_ring** command adds I/O guides to an existing I/O ring.

You can change the floorplan by adding I/O guides to the I/O ring. The I/O placer honors the new I/O ring guides during I/O placement.

The command returns 1 if the command succeeds, 0 if it fails, or TCL_ERROR if there is a command syntax error.

EXAMPLES

The following example adds guides to an I/O ring named "io_ring1"

```
prompt> add_to_io_ring io_ring1 { io_guide1 io_guide2 }
1
```

SEE ALSO

create_io_ring(2)
create_io_guide(2)
get_io_rings(2)
remove_from_io_ring(2)
report_io_rings(2)
remove_io_rings(2)

add_to_matching_type

Assigns cells, pins, and terminals to the matching type.

SYNTAX

int **add_to_matching_type**
matching_type
object_list

Data Types

matching_type list
object_list list

ARGUMENTS

matching_type

Specifies the name or collection of a single matching type. The matching type can be specified by using the **get_matching_types** command. However it must return a collection of one and only one matching type.

object_list

Specifies a list of cells, pins, and terminals to add to the matching type. The list can contain names, patterns, or collections. It is an error to add an object which is already in another matching type. Each object can exist in at most one matching type in the block.

DESCRIPTION

The **add_to_matching_type** command assigns cells, pins, and terminals to an existing matching type.

When I/O placement is performed, objects can be connected only to other objects with the same matching type. The matching type exists only in the block in which it was created, but its containing objects can span into subblocks. As a result, the matching types for the current block can apply to objects in subblocks when performing I/O placement from the current block, but matching types created in other blocks are not visible from the current block.

EXAMPLES

The following example assigns cells to a matching type named "myMatchingType1".

```
prompt> add_to_matching_type myMatchingType1 [get_cells inv1]
1
```

SEE ALSO

create_matching_type(2)
get_matching_types(2)
remove_from_matching_type(2)
remove_matching_types(2)
report_matching_types(2)

add_to_multisource_clock_sink_group

Adds clock sinks to a sink group defined for multisource clock tap assignment

SYNTAX

status **add_to_multisource_clock_sink_group**
-name *name*
-sinks *pin_or_port_list*

Data Types

name	string
pin_or_port_list	collection

ARGUMENTS

-name *name*

This required option defines the name of the sink group to which sinks are to be added. Sink groups are defined using the **create_multisource_clock_sink_group** command.

-sinks *pin_or_port_list*

This required option specifies a list of sinks that should get added to the sink list of the sink group given by option **-name**.

DESCRIPTION

The **add_to_multisource_clock_sink_group** command is used to add clock tree sinks to a sink group defined for multisource clock tap assignment.

The list of assigned sinks to a sink group can be obtained by accessing the sink group attribute **sinks**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This example adds the clock tree sink reg1/CK to sink group sink_group1.

```
prompt> add_to_multisource_clock_sink_group -name sink_group1 -sinks [get_pins reg1/CK]
```

SEE ALSO

create_multisource_clock_sink_group(2)
report_multisource_clock_sink_groups(2)
remove_multisource_clock_sink_groups(2)
get_multisource_clock_sink_groups(2)
remove_from_multisource_clock_sink_group(2)
synthesize_multisource_clock_taps(2)
set_multisource_clock_tap_options(2)

add_to_must_join_ports

Combines ports into a new or existing must-join port group.

SYNTAX

integer **add_to_must_join_ports**
port_list

Data Types

port_list collection

ARGUMENTS

port_list

Specifies a port name list or a port collection to form into a must-join group. Ports must be in the current block.

DESCRIPTION

The **add_to_must_join_ports** command combines the specified ports into a must-join port group. The command creates a new must-join port group if none of the ports in port_list are already in a must-join port group. If one of the ports is already in a must-join port group then the other ports will be added to the existing group. If two or more ports are already in must-join port groups then the groups will be merged into a single group and any unassociated ports will be added to the resulting group.

If the port list contains only a single port, the command will remove the port from any must-join port group.

EXAMPLES

The following example creates a must-join port group.

```
prompt> add_to_must_join_ports {A0 A1 A2}
1
```

The following example adds additional ports to the above must-join port group.

```
prompt> add_to_must_join_ports {A0 A3 A4 A5}
1
```

The following example remove a port from its must-join port group.

```
prompt> add_to_must_join_ports {A5}
1
```

add_to_net

Adds shapes, vias and via_matrixes to a net in the current design.

SYNTAX

integer **add_to_net**
net
shape_and_via_list

Data Types

net collection
shape_and_via_list collection

ARGUMENTS

net

Specifies the net to which to add shapes, vias and via_matrixes.

shape_and_via_list

Specifies the shapes, vias and via_matrixes to add to the net. The list may contain names, patterns, or collections. It is an error to include a shape, via or via_matrix which has already been assigned to another net. Text shapes may not be added to a net.

DESCRIPTION

The **add_to_net** command adds shapes, vias and via_matrixes to an existing net. Each shape, via and via_matrix may belong to one and only one net. Rects, polygons, paths, simple vias, simple array vias, and custom vias, via_matrixes may belong to a net. A text shape may not belong to any net.

EXAMPLES

The following example adds path shapes to a net.

```
prompt> add_to_net net1 [get_shapes PATH_16_*]
1
```

The following example adds simple vias to a net.

```
prompt> add_to_net net1 [get_vias VIA_S_*]
1
```

SEE ALSO

create_net(2)
get_nets(2)
remove_from_net(2)
create_shape(2)
get_shapes(2)
create_via(2)
get_vias(2)

add_to_net_bus

Adds the specified existing net to the existing net bus in the current design.

SYNTAX

collection **add_to_net_bus**
net_bus
net

Data Types

net_bus collection
net collection

ARGUMENTS

net_bus

Specifies the net bus in which net needs to be added.

net

Specifies the net to add to the net bus. The *net* should have the same base name as the *net_bus*.This net must be existing.

DESCRIPTION

This command adds the existing net to the existing net bus in the current design. Net bus numbering must increment in consecutive order; out-of-order bus nets are not allowed and will cause the tool to issue an error message.

The command returns the updated net bus(as a collection), an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

EXAMPLES

The following example creates a net bus named bus1, then adds net bus1[8] to the net bus named *bus1* having nets from 1 to 7. The third command tries to add an illegal bus net; bus net numbers must increment in consecutive order.

prompt> **create_net_bus -create_nets bus1[0:7]**

{bus1}

prompt> **add_to_net_bus bus1 bus1[8]**
{bus1}

prompt> **add_to_net_bus bus1 bus1[10]**
Error: Net addition will create a hole in the net bus. (NDM-120)

SEE ALSO

create_net_bus(2)
get_net_buses(2)
remove_from_net_bus(2)
remove_net_buses(2)
report_net_buses(2)

add_to_pin_blockage

Adds pins, nets, or ports to a pin blockage in the current design.

SYNTAX

status **add_to_pin_blockage**
pin_blockages
objects

Data Types

pin_blockages collection
objects collection

ARGUMENTS

pin_blockages

Specifies the pin blockages to which to add the specified objects.

objects

Specifies the physical pins, nets, or ports to add to the pin blockages. The specified objects must have the same type as the objects that already exist in the pin blockage.

DESCRIPTION

The **add_to_pin_blockage** command adds physical pins, nets, or ports to existing pin blockages. The specified objects can be added to multiple pin blockages.

Each pin blockage can contain objects of only one type (either pin, net, or port). It cannot contain a mixture of object types. If a pin blockage already contains pins, it cannot be assigned nets or ports. If a pin blockage already contains nets, it cannot be assigned pins or ports. If a pin blockage already contains ports, it cannot be assigned pins or nets.

EXAMPLES

The following example adds nets to a pin blockage named PIN_BLOCKAGE_1.

```
prompt> add_to_pin_blockage PIN_BLOCKAGE_1 [get_nets net*]
1
```

SEE ALSO

create_pin_blockage(2)
get_pin_blockages(2)
report_pin_blockages(2)
remove_pin_blockages(2)
remove_from_pin_blockage(2)
get_pins(2)
get_ports(2)
get_nets(2)

add_to_pin_guide

Adds pins, nets, or ports to a pin guide in the current design.

SYNTAX

status **add_to_pin_guide**
pin_guides
objects

Data Types

pin_guides collection
objects collection

ARGUMENTS

pin_guides

Specifies the pin guides to which to add the specified objects.

objects

Specifies the physical pins, nets (individual or bundled), or ports to add to the pin guides. The specified objects must have the same type as the objects that already exist in the pin guide. Currently bundles of supernets is not supported. If bundle of supernets is specified then it will be skipped with a warning message.

DESCRIPTION

The **add_to_pin_guide** command adds physical pins, nets, bundles of nets, or ports to existing pin guides. If the parent of a pin guide is the current design, ports or nets can be added to the pin guide. If the parent of a pin guide is a hierarchical cell, pins or nets can be added to the pin guide. Pins, nets, or ports can be added to multiple pin guides.

Each pin guide can contain objects of only one type, either pin, net, or port. It cannot contain a mixture of object types. If a pin guide already contains pins, it cannot be assigned nets or ports. If a pin guide already contains nets, it cannot be assigned pins or ports. If a pin guide already contains ports, it cannot be assigned pins or nets.

EXAMPLES

The following example adds nets to a pin guide named PIN_GUIDE_1.

```
prompt> add_to_pin_guide PIN_GUIDE_1 [get_nets net*]
1
```

SEE ALSO

create_pin_guide(2)
get_nets(2)
get_pin_guides(2)
get_pins(2)
get_ports(2)
remove_from_pin_guide(2)
remove_pin_guides(2)
report_pin_guides(2)

add_to_placement_attraction

Assigns cells to a placement_attraction in the current design.

SYNTAX

integer **add_to_placement_attraction**
placement_attraction_object
cell_list

Data Types

placement_attraction_object list
cell_list list

ARGUMENTS

placement_attraction_object

Specifies the name or collection of a single placement_attraction. The placement_attraction object can be specified by using the **get_placement_attractions** command. However it must return a collection of one and only one placement_attraction object.

cell_list

Specifies a list of cells to assign to the placement_attraction. The list may contain names, patterns, or collections. If a cell is hierarchical, its child cells are also implicitly in the placement_attraction, as well as any cells added to its hierarchy in the future. Cells that belong to a relative placement group cannot be individually added to a placement_attraction. All cells of a relative placement group must be added to a placement_attraction together.

DESCRIPTION

The **add_to_placement_attraction** command assigns cells to an existing placement_attraction.

You can change the floorplan by assigning cells to the placement_attraction. The placer honors the new placement_attraction cells during coarse placement.

EXAMPLES

The following example assigns cells to a placement_attraction named "placement_attraction1".

```
prompt> add_to_placement_attraction placement_attraction1 { inv1 add1 }
1
```

The following example assigns the hierarchical cell "mid" to the placement_attraction. The placement_attraction implicitly contains all cells within "mid".

```
prompt> add_to_placement_attraction placement_attraction1 [get_cells mid]
1
```

The following example assigns all cells of a relative placement group "rp1" to the placement_attraction.

```
prompt> add_to_placement_attraction placement_attraction1 [get_cells -of_objects rp1]
1
```

SEE ALSO

create_placement_attraction(2)
get_placement_attractions(2)
remove_placement_attractions(2)
remove_from_placement_attraction(2)
report_placement_attractions(2)

add_to_port_bus

Adds the specified existing port to the existing port bus in the current design.

SYNTAX

collection **add_to_port_bus**
port_bus
port

Data Types

port_bus collection
port collection

ARGUMENTS

port_bus

Specifies the port bus in which port needs to be added.

port

Specifies the port to add to the port bus. The *port* should have the same base name as the *port_bus*.This port must be existing.

DESCRIPTION

This command adds the existing port to the existing port bus in the current design. Port bus numbering must increment in consecutive order; out-of-order bus ports are not allowed and will cause the tool to issue an error message.

The command returns the updated port bus(as a collection), an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

EXAMPLES

The following example creates a port bus named bus1, then adds port bus1[8] to the port bus named *bus1* having ports from 1 to 7. The third command tries to add an illegal bus port; bus port numbers must increment in consecutive order.

prompt> **create_port_bus -create_ports bus1[0:7]**

{bus1}

prompt> **add_to_port_bus bus1 bus1[8]**
{bus1}

prompt> **add_to_port_bus bus1 bus1[10]**
Error: Port addition will create a hole in the port bus. (NDM-115)

SEE ALSO

create_port_bus(2)
get_port_buses(2)
remove_from_port_bus(2)
remove_port_buses(2)
report_port_buses(2)

add_to_routing_corridor

Associates nets, supernets, and their bundles with a routing corridor.

SYNTAX

status **add_to_routing_corridor**
routing_corridor
object_list

Data Types

routing_corridor list
object_list list

ARGUMENTS

routing_corridor

Specifies the routing corridor with which to associate the specified nets.

This is a required option.

object_list

Specifies the nets and supernets (individual or bundled) to associate with the routing corridor.

This is a required option.

DESCRIPTION

This command associates nets, supernets, and their bundles with a routing corridor.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example associates the nets named net_a and net_b with the routing corridor named corridor_a.

```
prompt> add_to_routing_corridor \
  corridor_a [get_nets [list net_a net_b]]
```

The following example associates the supernet named supernet_1 with the routing corridor named corridor_a

```
prompt> add_to_routing_corridor \
  corridor_a supernet_1
```

SEE ALSO

check_routing_corridors(2)
create_routing_corridor(2)
create_routing_corridor_shape(2)
get_nets(2)
get_routing_corridors(2)
remove_from_routing_corridor(2)
remove_routing_corridors(2)
report_routing_corridors(2)

add_to_rp_group

Adds cell, relative placement group or relative placement blockage to a relative placement group in the current design.

SYNTAX

integer **add_to_rp_group**
rp_group_object
[-row *row_number*]
[-column *column_number*]
[-override_alignment *left* | *right*]
-cells *cell_list*
[-num_rows *number_of_rows*]
[-num_columns *number_of_columns*]
[-pin_name *pin_name*]
[-orientation *direction*]
[-free_placement]
[-height *object_height*]
[-width *object_width*]|
-rp_group *rp_group_object* |
-blockage *blockage_name*
[-width *object_width*]
[-height *object_height*]
[-allow_overlap]

Data Types

rp_group_object collection
row_number integer
column_number integer
left string
right string
cell_list list
number_of_rows integer
number_of_columns integer
pin_name string
direction list
object_height integer
object_width integer
blockage_name string

ARGUMENTS

rp_group_object

Specifies the name or collection of a single relative placement group, to which the objects will be added. The relative placement object can be specified by using the **get_rp_groups** command. However it must return a collection of one and only one relative placement group.

-row ***row_number***

Specifies the row position in which to add the object. If you do not specify the row position, the default is zero.

-column ***column_number***

Specifies the column position in which to add the object. If you do not specify the column position, the default is zero.

-override_alignment ***left*** | ***right***

Specifies the type of the override alignment for the specified relative placement object in its column. The specified override alignment will override the alignment type that was inherited from its parent relative placement group. The right override alignment is not supported for the multi column relative placement cell.

-cells ***cell_list***

Specifies a list of cells to add to the relative placement group. If this command option is used with *free_placement* command option then all the cells specified in the list will be added otherwise, only the first cell of the list will be added to the RP group location. The list may contain names, patterns, or collections. It is an error to include a cell, which has already been assigned to another relative placement group. The hard macro cells can also be added to RP group.

-num_rows ***number_of_rows***

Specifies the number of rows to which you add the cell(s). The default is 1 if you do not specify this option. This option can also be used with *-free_placement* for adding free placement cells.

-num_columns ***number_of_columns***

Specifies the number of columns to which you add the cell(s). The default is 1 if you do not specify this option. The pin alignment is not supported for a cell that occupies multiple columns. This option can also be used with *-free_placement* for adding free placement cells.

-pin_name ***pin_name***

Specifies the name of the pin to use for pin alignment of this cell with other cells in a group. This overrides the default pin name specified for the relative placement group into which it is being added. This option can only be used when adding a leaf cell with the **-cells** option. The option is not applicable on hard macro cells.

-orientation ***direction***

Specifies the placement orientation of the cell, with respect to the group in which the cell is being added. You can specify a list of possible orientations, and the tool chooses the first legal orientation for the cell. The option is not applicable on hard macro cells.

-free_placement

Specifies that the given cells will not have a fixed relative location inside the relative placement group. The default value is false and it can be used only with the -cells option. This option is used along with *-num_rows*, *-num_columns*, *-height* and *-width* options. The *-num_rows* and *-num_columns* options will specify number of rows and columns to which free placement cells will be added. The *-height* and *-width* options will specify the height and width of the free placement region.

-height ***object_height***

Specifies the height of the blockage being added when used with the **-blockage** option or the height of the free placement region when used with **-free_placement**. option. The unit for height is the height of the site row for a particular library. If you do not specify the height, it defaults to the site row height.

-width ***object_width***

Specifies the width of the blockage being added when used with **-blockage** or the width of the free placement region when used with **-free_placement**. The unit for width is the width of the site row for a particular library. If you do not specify the width, it defaults to the width of the site for horizontal_compression **tiling_type** and for other tiling types it is the width of the column into which the blockage is being added. If you do not specify the width with **-free_placement** option, the width of free placement region defaults to the width of the site row.

-rp_group ***hier_rp_group_object***

Specifies the relative placement group name or collection to add to the relative placement group. It is an error to include a relative placement group, which has already been added.

-blockage ***blockage_name***

Specifies the name of the blockage to add to the relative placement group. It is an error to provide a *blockage_name*, which has already been used for some other relative placement blockage.

-allow_overlap

Specifies that the blockage can be placed over fixed objects. It can also overlap with relative placement blockages, but cannot overlap with the relative placement cells. The blockage cannot overlap with fixed objects if the option is not given.

DESCRIPTION

The **add_to_rp_group** command adds cells, relative placement group or relative placement blockage to existing relative placement group. The added object is placed in a particular lattice position of the group as specified by a **row** and **column** options. The hard macro cells also can be added to the relative placement group. The user has to follow a flow when hard macros are present in the relative placement group. Please refer to user guide for more details.

EXAMPLES

The following example uses **add_to_rp_group** to add a cell to an existing relative placement group and then adds that group hierarchically to another existing group.

```
prompt> get_rp_groups grp_ripple
{grp_ripple}

prompt> add_to_rp_group grp_ripple -cells carry_in_1 -row 1 -column 2
1

prompt> add_to_rp_group top_group -rp_group grp_ripple -row 2 -column 2
1
```

The following example uses **add_to_rp_group** to add a blockage to an existing relative placement group.

```
prompt> add_to_rp_group grp_ripple -blockage grp_ripple_blkg2 \
  -row 2 -column 2 -allow_overlap
1
```

The following example uses **add_to_rp_group** to add free placement cells to an existing relative placement group.

```
prompt> add_to_rp_group grp_ripple \
  -cells {carry_in_1 carry_in_2 carry_out_1} -free_placement \
```

-row 1 -column 1 -height 2 -width 5 -num_rows 2 -num_columns 2

SEE ALSO

create_rp_group(2)
get_rp_groups(2)
remove_from_rp_group(2)
write_rp_groups(2)
remove_rp_groups(2)

add_to_scan_chain

Adds the specified existing stub_chains to the existing scan_chain in the current design.

SYNTAX

collection **add_to_scan_chain**
[-at *index*]
scan_chain
stub_chains

Data Types

index int
scan_chain collection
stub_chains collection

ARGUMENTS

index

Specifies the position within the scan_chain to add the stub_chains.

scan_chain

Specifies the name or collection of the scan chain to which to add stub_chains.

stub_chains

Specifies the name or collection of the stub chains to add into the *scan_chain*.

DESCRIPTION

This command adds the existing *stub_chains* into the existing *scan_chain*. The index value can range from 0 to the number of stub chains in the *scan_chain* minus 1. If *index* is not specified, default position is the end of *scan_chain*.

The command returns 1 if the command succeeds, 0 if it fails, or TCL_ERROR if there is a command syntax error.

EXAMPLES

The following example creates a stub_chain named stub1, a scan_chain named scan1 and then adds the stub_chain stub1 to the scan_chain scan1 at the default index.

```
prompt> create_stub_chain -name stub1
{stub1}

prompt> create_scan_chain -name scan1
{scan1}

prompt> add_to_scan_chain scan1 stub1
1
```

The following example creates 3 stub_chains {stub1, stub2, stub3}, a scan_chain named scan1 having stub_chains {stub1, stub3} and then adds stub_chain stub3 at index 1 into the scan_chain.

```
prompt> create_stub_chain -name stub1
{stub1}

prompt> create_stub_chain -name stub1
{stub2}

prompt> create_stub_chain -name stub1
{stub3}

prompt> create_scan_chain -name scan1 {stub1 stub3}
{scan1}

prompt> add_to_scan_chain -at 1 scan1 stub2
1
```

SEE ALSO

create_scan_chain(2)
get_scan_chains(2)
remove_from_scan_chain(2)
remove_scan_chains(2)
report_scan_chains(2)

add_via_mapping

Adds via mapping options for redundant via insertion.

SYNTAX

status **add_via_mapping**
-from *via_pattern*
-to *list_of_destination_vias*
[-weight *weight*]
[-transform all | flip | none | rotate]
[-force]
| -from_icc_file *script*

Data Types

via_pattern	string
list_of_destination_vias	list
weight	integer
script	string

ARGUMENTS

-from *via_pattern*

Specifies the vias to be replaced during redundant via insertion.

A via pattern has two parts: viaDefName and site_spec mxn. viaDefName is via definition name. MxN specifies the number of contacts in the horizontal (M) and vertical (N) directions.

This option supports simple vias.

This is a required option and must be specified.

-to *list_of_destination_vias*

Specifies the list of vias to insert during redundant via insertion. This is a required option and must be specified.

This option supports simple vias and custom vias. If a custom via is specified, the target MxN dimension must be 1x1.

-weight *weight*

Specifies the weight of contacts for the vias inserted during redundant via insertion. The weight is an integer between 1 and 30 (inclusive). Via mappings with a higher weight are preferred over via mappings with a lower weight.

The default is 1 for all via mappings.

-transform all | **flip** | **none** | **rotate**

Specifies the types of rotated via arrays that can be used during signal routing and redundant via insertion.

The definition for each mode is:

- **none** : Do not rotate or swap via arrays.
- **flip** : Use 1 x N and N x 1 via arrays as rotated equivalent vias.
- **rotate** : Rotate line via arrays instead of flipping (swapping row and column numbers).
- **all** : Use all four possible via arrays. This option is the default.

-force

Overwrites via mappings with the same "from" and "to" via patterns, if it already exists.

-from_icc_file ***script***

Specifies the script file that contains via mapping settings from IC Compiler **define_zrt_redundant_vias** command.

Before apply the via mapping settings in the script file, all existing via mappings will be removed first.

This option is exclusive with all other options.

DESCRIPTION

This command sets options for redundant via insertion. The redundant via definitions are saved in the design.

This command can also accept a script file that contains the **define_zrt_redundant_vias** command from IC Compiler.

EXAMPLES

The following example sets the options for via replacement with double vias:

```
prompt> add_via_mapping  -from {VIA12 1x1} \
  -to { VIA12 1x2}
```

The following example sets the options for via replacement with weight of 2:

```
prompt> add_via_mapping -from {VIA23_r90 1x1}  \
  -to { VIA23 2x1} -weight 2
```

The following example sets the options for via replacement by a script file:

```
prompt> add_via_mapping -from_icc_file RVI.tcl
```

The RVI.tcl has following content:

```
set VIA_OPT_LIST {
  {"VIA23"         "VIA23"       2     1      2}
  {"VIA23_VV"      "VIA23_VV"    2     1      2}
  {"VIA23"         "VIA23_FBD"   1     1      9}
```

```
    {"VIA23"         "VIA23_FBS"    1     1      8}
    {"VIA23"         "VIA23_PBD"    1     1      7}
    {"VIA23_LONG_V"    "VIA23_PBD"    1     1      8}
    {"VIA23_LONG_H"    "VIA23_FBD"    1     1      9}
}
set _from_list ""
set _to_list ""
set _xfactor_list ""
set _yfactor_list ""
set _to_via_weight_list ""

foreach {_from _to _xfactor _yfactor _to_via_weight} [join $VIA_OPT_LIST] {
    set _from_list           "$_from_list          $_from"
    set _to_list             "$_to_list            $_to"
    set _xfactor_list        "$_xfactor_list       $_xfactor"
    set _yfactor_list        "$_yfactor_list       $_yfactor"
    set _to_via_weight_list    "$_to_via_weight_list   $_to_via_weight"
}
define_zrt_redundant_vias \
-from_via      $_from_list \
-to_via        $_to_list \
-to_via_x_size  $_xfactor_list \
-to_via_y_size  $_yfactor_list \
-to_via_weights $_to_via_weight_list
```

SEE ALSO

remove_via_mappings(2)
report_via_mapping(2)

alias

Creates a pseudo-command that expands to one or more words, or lists current alias definitions.

SYNTAX

string **alias**
[*name*]
[*def*]

Data Types

name string
def string

ARGUMENTS

name

Specifies a name of the alias to define or display. The name must begin with a letter, and can contain letters, underscores, and numbers.

def

Expands the alias. That is, the replacement text for the alias name.

DESCRIPTION

The **alias** command defines or displays command aliases. With no arguments, the **alias** command displays all currently defined aliases and their expansions. With a single argument, the **alias** command displays the expansion for the given alias name. With more than one argument, an alias is created that is named by the first argument and expanding to the remaining arguments.

You cannot create an alias using the name of any existing command or procedure. Thus, you cannot use **alias** to redefine existing commands.

Aliases can refer to other aliases.

Aliases are only expanded when they are the first word in a command.

EXAMPLES

Although commands can be abbreviated, sometimes there is a conflict with another command. The following example shows how to use **alias** to get around the conflict:

```
prompt> alias q quit
```

The following example shows how to use **alias** to create a shortcut for commonly-used command invocations:

```
prompt> alias include {source -echo -verbose}

prompt> alias rt100 {report_timing -max_paths 100}
```

After the previous commands, the command **include script.tcl** is replaced with **source -echo -verbose script.tcl** before the command is interpreted.

The following examples show how to display aliases using **alias**. Note that when displaying all aliases, they are in alphabetical order.

```
prompt> alias rt100
rt100  report_timing -max_paths 100

prompt> alias
include  source -echo -verbose
q  quit
rt100  report_timing -max_paths 100
```

SEE ALSO

unalias(2)

align_objects

Aligns specified objects.

SYNTAX

status **align_objects**
[-anchor *object_list*]
[-parent]
[-to *point*]
[-to_box *box*]
[-side *alignment_side*]
[-anchor_side *anchor_side*]
[-offset *offset*]
[-skip_anchor]
[-group]
[-margin]
[-force]
[-simple]
[*object_list*]

Data Types

object_list collection
point float
box list
alignment_side string
anchor_side string
offset float

ARGUMENTS

object_list

Collection or selection set containing objects to align. If this option is not specified, the global selection set is used.

-anchor *object_list*

Specifies the objects to which other objects are aligned.

-parent

Aligns objects to the parent edge.

For terminals and I/O pads, the parent object is the die. For soft macro pins, the parent object is the soft macro. For all other objects, the parent is the core.

-to *point*

Specifies a point with which to align the objects.

-to_box *box*

Specifies a box with which to align the objects.

-side *alignment_side*

Specifies the side of the object to be aligned. The valid value is one of **left**, **right**, **hcenter**, **bottom**, **top**, **vcenter**.

The description of the valid values are as follows:

left - align to the left side
right - align to the right side
top - align to the top side
bottom - align to the bottom side
hcenter - align to the horizontal center
vcenter - align to the vertical center

The default is **left**.

-anchor_side *anchor_side*

Specifies the anchor object side to use when aligning objects. Valid values are **auto**, **ll**, **ur**, or **center**.

auto - determine anchor side automatically
ll - align to the lower-left side of the anchor object
ur - align to the upper-right side of the anchor object
center - align to the center of the anchor object

The default is auto.

-offset *offset*

Specifies the offset from the specified anchor object's side. The default is **0.0**.

-skip_anchor

When **-offset** is specified and anchor is not given explicitly with -anchor/-parent/-to/-to_box options then this option allows to skip implicit anchor object when applying offset. This option is not compatible with -anchor, -parent, -to or -to_box options and requires -offset option be specified.

-group

Treat the objects as a single object. If you specify **-group**, the tool maintains the relative position between the specified objects. Be default, objects are treated individually.

-margin

Use extended objects boundaries. When you specify **-margin**, the block boundaries are extended with keepout margins and voltage area boundaries are extended with guard bands. To exclude certain block keepout margins from consideration, use the following command:

```
prompt> win_set_filter -class cell \
  -filter {hard_macro_margin hard_margin route_blockage_margin soft_margin}
```

-force

Resizes locked or fixed objects. By default, such objects are not resized according to global edit settings. Edit settings can be accessed using **get_edit_setting**.

-simple

Disable snapping and editing constraints. By default, objects are snapped according to global snap settings. Snap settings can be accessed using **get_snap_setting**.

DESCRIPTION

This command aligns a list of unfixed objects so that their left, right, top, or bottom edges are coincident with, or at a specified offset from, an anchor object or anchor position.

You can specify an anchor object or anchor position, or allow the command to automatically determine the anchor object from the supplied list of objects by using the following algorithm:

- If any objects are marked as fixed, the anchor object is selected from the set of fixed objects. The rightmost fixed object is used for right alignment, the topmost fixed object is used for top alignment, and so on.
- If no objects are marked as fixed, then the anchor object is the leftmost, rightmost, topmost, or bottommost object for alignment of left, right, top, and bottom, respectively.

NOTE:Snapping is done automatically using global snap settings.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example moves the currently selected objects so that their left edges are *20* units from the left edge of the core.

```
prompt> align_objects -parent -offset 20
```

The alternative syntax uses a collection to specify the objects.

```
prompt> align_objects [get_selection] -parent -offset 20
```

SEE ALSO

change_selection(2)
distribute_objects(2)
get_edit_setting(2)
get_selection(2)
get_snap_setting(2)
set_edit_setting(2)
set_snap_setting(2)
snap_objects(2)
spread_objects(2)

align_pins

Aligns the specified block pins with their connected pins. If no block pins are specified, the command aligns the currently selected block pins.

SYNTAX

status **align_pins**
[*object_list*]
[-change_layer_width]
[-change_layer_height]

Data Types

object_list collection

ARGUMENTS

object_list

Specifies a collection or selection list that contains pins to align. If this option is not specified, the global selection list is used.

-change_layer_width

Changes the pins' layer and width to match the layer and width of the connected pin.

-change_layer_height

Changes the pins' layer and height to match the layer and width of the connected pin.

DESCRIPTION

This command aligns pins specified in *object_list* or the currently selected pins. Pins are aligned with the pins to which they connect based on the current snapping constraints. For each pin to align, the tool finds another pin that connects to the same net and uses the connected pin as an anchor object. After the command completes, the center points for both the selected pin and the anchor pin should be on the same horizontal or vertical lines if both pins satisfy the current snapping constraints.

Note that the **align_pins** command does not consider side, layer, offset, and other pin constraints specified by the **set_individual_pin_constraints**, **set_block_pin_constraints**, or **read_pin_constraints** commands. To place pins while honoring constraints set by these commands, use the **place_pins** command.

You should only use the **align_pins** command when the connected pins are physically close and have no obstacles between them. This command does not perform overlap checking or blockage avoidance. Alignment is not performed and the pin is not moved if

any of the following are true:

- Pins are marked as fixed
- Selected objects are not of type pin, port, or terminal
- Pins belong to a power or ground net
- Pins have multiple terminals that do not intersect

Note that snapping is done automatically based on global snap settings.

EXAMPLES

The following example moves the currently selected pins so that the pins align either horizontally or vertically with their connected pins.

```
prompt> align_pins
```

The following example aligns the currently selected pin and changes the layer and width of the selected pin to match the layer and width of the connected pin.

```
prompt> align_pins -change_layer_width
```

The following example uses an alternative syntax to specify a collection of objects.

```
prompt> align_pins [get_pins I_ALU/OUT[2]]
```

SEE ALSO

align_objects(2)
change_selection(2)
distribute_objects(2)
get_edit_setting(2)
get_selection(2)
get_snap_setting(2)
set_edit_setting(2)
set_snap_setting(2)
snap_objects(2)

all_clocks

Creates a collection of all clocks in the specified design.

SYNTAX

collection **all_clocks**
[-design *design*]
[-mode *mode*]

Data Types

design collection
mode collection

ARGUMENTS

-design *design*

Specifies the design for which to return the clocks. You can specify only a single design.

If you do not specify this option, the command returns all clocks in the current design.

-mode *mode*

Specifies the mode for which to return the clocks. You can specify only a single mode.

If you do not specify this option, the command returns all clocks in the current mode.

DESCRIPTION

The **all_clocks** command creates a collection of all clocks in the specifed design. If you did not define any clocks, the empty collection (empty string) is returned.

If you want only certain clocks, use the **get_clocks** command to create a collection of clocks that match a specific pattern and optionally pass in filter criteria.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-mode** option.

EXAMPLES

The following example applies the **set_propagated_clock** command to all clocks in the design.

```
prompt> set_propagated_clock [all_clocks]
```

SEE ALSO

collections(2)
create_clock(2)
get_clocks(2)
set_propagated_clock(2)

all_connected

Creates a collection of objects connected to nets, scalar nets of net_bus, pins, ports or scalar ports of port_bus.

SYNTAX

collection **all_connected**
[-leaf]
[-physical_context]
objects

Data Types

objects list

ARGUMENTS

-leaf

Specifies that only leaf pins are returned for a hierarchical net.

For nonhierarchical nets, there is no difference in output.

-physical_context

Searches physical objects rather than logical objects.

When specified for nets, the command returns physical pins or ports. For pins and ports, the command returns the physical net, which might be in a different hierarchy and connected to the pin through hierarchical pins or ports.

objects

Specifies the objects whose connections are returned. You must specify objects that are nets, net_bus, pins, ports or port_bus.

This is a required argument.

DESCRIPTION

The **all_connected** command creates a collection of objects connected to the specified nets, net_buses, pins, ports or port_buses. The collection can contain nets, ports, pins, or a combination of ports and pins. In the latter case, the ports are first, followed by the pins.

When issued from the command prompt, the **all_connected** command behaves as though the **query_objects** command has been

called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **shell.common.collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example shows all objects connected to the net named CLOCK.

```
prompt> query_objects -verbose [all_connected [get_nets CLOCK]]
{{port CLOCK} {pin U1/CP} {pin U2/CP} {pin U3/CP} {pin U4/CP}}
```

The following example shows all objects connected to the net_bus N.

```
prompt> all_connected N
{{P[7]} {P[6]} {P[5]} {P[4]} {P[3]} {P[2]} {P[1]} {P[0]}}
```

SEE ALSO

collections(2)
get_nets(2)
get_pins(2)
get_ports(2)
query_objects(2)
shell.common.collection_result_display_limit(3)

all_corners

Creates a collection of all corners in the current design. You can assign these corners to a variable or pass them into another command.

SYNTAX

collection **all_corners**
[-design *design*]

Data Types

design collection

ARGUMENTS

-design *design*

Name of the top design. The default is the current design.

DESCRIPTION

The **all_corners** command creates a collection of all corners in the current design. If you do not define any corners, a collection containing the default corner is returned.

If you want only certain corners, use **get_corners** to create a collection of corners matching a specific pattern and optionally pass in filter criteria.

Multicorner-Multimode Support

This command works on all corners.

SEE ALSO

collections(2)
create_corner(2)
current_corner(2)
get_corners(2)

all_exceptions

Creates a collection of all timing exceptions in the current design.

SYNTAX

collection **all_exceptions**
[-design *design*]
[-mode *mode*]

Data Types

design	collection
mode	collection

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-mode *mode*

Specifies the mode for finding objects. If this is not specified, objects will be found in the current mode.

DESCRIPTION

The **all_exceptions** command creates a collection of all timing exceptions in the current scenario. If no timing exceptions exist, the empty string is returned. You can assign these exceptions to a variable or pass them into another command. Timing exceptions are created by **set_false_path**, **set_multicycle_path**, **set_max_delay** or **set_min_delay**.

If you want only certain exceptions, use **get_exceptions** to create a collection of exceptions matching a specific pattern and optionally pass in filter criteria.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-mode** option.

EXAMPLES

This example sets the variable allExcepts to a collection of all timing exceptions in the design.

```
prompt> set allExcepts [all_exceptions]
```

SEE ALSO

collections(2)
set_false_path(2)
set_multicycle_path(2)
set_max_delay(2)
set_min_delay(2)
get_exceptions(2)
shell.common.collection_result_display_limit(3)

all_fanin

Creates a collection of pins, ports, or cells in the fanin of the specified sinks.

SYNTAX

collection **all_fanin**
[-from *from_list*]
[-through *through_list*]
-to *to_list*
[-flat] [-only_cells]
[-quiet]
[-startpoints_only]
[-levels *level_count*]
[-pin_levels *pin_count*]
[-step_into_hierarchy]
[-trace_arcs timing | enabled | all]
[-continue_trace *pin_types*]

Data Types

from_list list
through_list list
to_list list
level_count integer
pin_count integer

ARGUMENTS

-from *from_list*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. The timing fanin of each object in *to_list* becomes part of the resulting collection only if it is in the fanout cone of at least one object in *from_list*. If a net is specified, the effect is the same as listing all load pins on the net.

-through *through_list*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. If a *through_list* is specified the fanin of each object in *to_list* is restricted to objects on paths through the pins, ports or nets in *through_list*.

-to *to_list*

Specifies a list of sink pins, ports, or nets to trace. Each object is a named pin, port, or net or a collection of pins, ports, or nets. The timing fanin of each sink becomes part of the resulting collection. If a net is specified, the effect is the same as listing all driver pins on the net.

This option is required.

-flat

Includes objects throughout the design hierarchy in the result. This means that the only non-leaf objects in the result are hierarchical sink pins.

If you do not specify this option, the result includes only objects within the same hierarchical level as the current sink.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-only_cells

Includes only cells in the timing fanin of the specified sinks in the result and not pins or ports.

-startpoints_only

Includes only the timing startpoints in the result.

-levels ***level_count***

Stops the traversal when it reaches the specified cell depth, where the counting is performed over the layers of cells of the same distance from the sink. This option cannot be used together with **-pin_levels**.

-pin_levels ***pin_count***

Stops the traversal when it reaches the specified pin depth, where the counting is performed over the layers of pins of the same distance from the sink. This option cannot be used together with **-levels**.

-step_into_hierarchy

Performs counting as though the design is flat. Although pins inside the hierarchy are not returned, they determine the depth of the related output pins.

If you do not specify this switch, a hierarchical block at the same level of hierarchy as the current sink is considered to be a cell; the input pins are considered a single level away from the related output pins, regardless of what is inside the block.

This option cannot be used with the **-flat** option and only has effect with either the **-levels** or **-pin_levels** option.

-trace_arcs timing | enabled | all

Specifies the type of combinational arcs to trace during the traversal.

Allowed values are **timing** (the default), which permits tracing only of valid timing arcs (that is, arcs which are neither disabled nor invalid due to case analysis); **enabled**, which permits the tracing of all enabled arcs and disregards case analysis values; and **all**, which permits the tracing of all combinational arcs regardless of either case analysis or arc disabling.

-continue_trace ***pin_types***

This option permits the user to override the default behavior of the traversal which is to stop at all timing startpoints. When used, the travsal is permitted through startpoint pins of a type specified with *pin_types*. Allowed *pin_types* are:

- **generated_clock_source** - Specifies that the traversal is to continue tracing through source pins of generated clocks including sequential clock pins in the generated clock source network.

DESCRIPTION

The **all_fanin** command creates a collection of objects in the timing fanin of the specified sink pins, ports, or nets in the design. A pin is considered to be in the timing fanin of a sink if there is a timing path through combinational logic from the pin to that sink (see also the **-trace_arcs** option). The fanin stops at the clock pins of registers (sequential cells).

If the current instance in the design is not the top level of hierarchy, only objects within the current instance are returned.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

This example shows the timing fanin of a port in the design. The fanin includes a register, reg1.

```
prompt> query_objects [all_fanin -to out_1]
{"out_1", "reg1/Q", "reg1/CP"}
```

This example shows the flat mode of **all_fanin**. The sink is an input pin of a hierarchical cell, H1, which is connected to an output pin of another hierarchical cell, H2. H2 contains additional hierarchy and eventually, a leaf cell with two inputs, each of which has a top-level register in its fanin.

```
prompt> query_objects [all_fanin -to H1/a -flat]
{"H1/a", "H2/U1/n1/Z", "H2/U1/n1/A", "H2/U1/n1/B",
 "reg1/Q", "reg2/Q", "reg1/CP", "reg2/CP"}
```

This example shows the timing fanin of an output port that passes through pin u1/z or pin u2/z.

```
prompt> query_objects [all_fanin -to out -through {u1/z u2/z}]
{"out", "u3/z", "u3/a", "u3/b", "u1/z", "u1/a", "u2/z", "u2/z",
 "reg1/q", "reg2/q", "reg1/cp", "reg2/cp"}
```

This example shows the timing fanin of an output port the passes through pin u1/z and pin u3/a.

```
prompt> query_objects [all_fanin -to out -through u1/z -through u3/a]
{"out", "u3/z", "u3/a", "u1/z", "u1/a", "reg1/q", "reg1/cp"}
```

SEE ALSO

all_fanout(2)
report_transitive_fanin(2)
current_instance(2)

all_fanout

Creates a collection of pins/ports or cells in the fanout of the specified sources.

SYNTAX

collection **all_fanout**
-from *from_list*
[-through *through_list*]
[-to *to_list*]
[-clock_tree]
[-flat]
[-quiet]
[-only_cells]
[-endpoints_only]
[-levels *level_count*]
[-pin_levels *pin_count*]
[-trace_arcs *arc_types*]
[-step_into_hierarchy]
[-continue_trace *pin_types*]

Data Types

from_list list
through_list list
to_list list
level_count integer
pin_count integer
arc_types string
pin_types string

ARGUMENTS

-from *from_list*

Specifies a list of source pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. The timing fanout of each source in *from_list* becomes part of the resulting collection. If a net is specified, the effect is the same as listing all load pins on the net. This option is exclusive with the **-clock_tree** option.

-through *through_list*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. If a *through_list* is specified the fanout of each object in *from_list* is restricted to pins on paths through the pins, ports or nets in *through_list*.

-to *to_list*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. The timing fanout of each object in *from_list* becomes part of the resulting collection if it is in the fanin cone of at least one object in *to_list*. If a net is specified, the effect is the same as listing all driver pins on the net.

-clock_tree

Indicates that all clock source pins or ports in the design are to be used as the list of sources. Clock sources are specified using **create_clock**. If there are no clocks, or if the clocks have no sources, the result is the empty collection. This option is exclusive with the **-from** option.

-flat

There are two major modes in which **all_fanout** functions: hierarchical (the default) and flat. When in hierarchical mode, only objects within the same hierarchical level as the current source are included in the result. In flat mode, the only non-leaf objects in the result will be hierarchical source pins.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-only_cells

The result will include only cells in the timing fanout of the *from_list* and not pins or ports.

-endpoints_only

When this option is specified, only the timing endpoints will be included in the result.

-levels *level_count*

The traversal will stop when reaching a depth of search of *cell_count* hops, where the counting is performed over the layers of cells of same distance from the source. This option cannot be used together with **-pin_levels**.

-pin_levels *pin_count*

The traversal will stop when reaching a depth of search of *pin_count* hops, where the counting is performed over the layers of pins of same distance from the source. This option cannot be used together with **-levels**.

-trace_arcs *arc_types*

Specifies the type of combinational arcs to trace during the traversal. Allowed values are **timing** (the default), which permits tracing only of valid timing arcs (that is, arcs which are neither disabled nor invalid due to case analysis); **enabled**, which permits the tracing of all enabled arcs and disregards case analysis values; and **all**, which permits the tracing of all combinational arcs regardless of either case analysis or arc disabling.

-step_into_hierarchy

This option may only be used in hierarchical mode and only has effect with either **-levels** or **-pin_levels**. Without the switch, a hierarchical block at the same level of hierarchy as the current sink is considered to be a cell; the output pins are considered a single level away from the related input pins, regardless of what is inside the block. With the switch enabled, the counting is performed as though the design were flat, and although pins inside the hierarchy are not returned, they determine the depth of the related input pins.

-continue_trace *pin_types*

This option permits the user to override the default behavior of the traversal which is to stop at all timing endpoints. When used, the travsal is permitted through endpoint pins of a type specified with *pin_types*. Allowed *pin_types* are:

- **generated_clock_source** - Specifies that the traversal is to continue tracing through source pins of generated clocks including sequential clock pins in the generated clock source network.

DESCRIPTION

The **all_fanout** command creates a collection of objects in the timing fanout of specified source pins/ports or nets in the design. A pin is considered to be in the timing fanout of a source if there is a timing path through combinational logic from that source to the pin (please also see the **-trace_arcs** option). The fanout stops at the inputs to registers (sequential cells). The sources are specified using either **-clock_tree** or **-from** *from_list*.

If current instance in the design is not the top level of hierarchy, only objects within the current instance will be returned.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

This example shows the timing fanout of a port in the design. The fanout includes a register, reg3.

```
prompt> query_objects [all_fanout -from in1]
{"in1", "reg3/D"}
```

This example shows the difference between the hierarchical and flat modes of **all_fanout**. The source is an output pin of a hierarchical cell, H3/z1, which is connected to an input pin of another hierarchical cell, H4/a. H4 contains a leaf cell U1 with input A and output Z. The first command is hierarchical mode, and shows that hierarchical pins are included in the result. The second command is in leaf mode, and leaf pins from the lower level are included.

```
prompt> query_objects [all_fanout -from H3/z1]
{"H3/z1", "H4/a", "H4/z", "reg2/D"}

prompt> query_objects [all_fanout -from H3/z1 -flat]
{"H3/z1", "H4/U1/A", "H4/U1/Z", "reg2/D"}
```

This example shows how to get the fanout of reg1/CP on paths passing through u1/Z and U3/Z.

```
prompt> query_objects [all_fanout -from reg1/CP -through U1/Z \
        -through U3/A]
{"out", "U3/Z", "U3/A", "U1/Z", "U1/A", "reg1/Q", "reg1/CP"}
```

SEE ALSO

all_fanin(2)
current_instance(2)
report_transitive_fanin(2)

all_high_transitive_fanout

Returns a collection of high-fanout nets from the current design or from the specified input collection. A tie net will be filtered out in the collection.

SYNTAX

collection **all_high_transitive_fanout**
-nets
[-threshold *value*]
[*input_coll*]
[-through_buf_inv]

Data Types

value float
input_coll collection

ARGUMENTS

-nets

Returns the collection of high-fanout nets.

-threshold *value*

Specifies a threshold value used to determine if a net is a high-fanout net. The *value* is a user-specified value. If the value is less than 1, a default value will be the threshold. By default, the value of the **design.high_fanout_net_threshold** application option is used to determine if a net is a high-fanout net.

input_coll

Searches the specified input collection for high-fanout nets. Objects are to be searched only from the contents of the specified input collection, rather than from the design.

-through_buf_inv

Indicates to treat a buffer tree as transparent. The leaf loads of the buffer tree are treated as the fanouts of the net.

DESCRIPTION

The **all_high_transitive_fanout** command returns a collection of all high-fanout nets in the design. The tool searches the entire design hierarchy. It returns nets from the entire design, not just from the top level.

To determine if a net is a high-fanout net, use the **design.high_fanout_net_threshold** option value as the threshold value. The command returns all nets with a fanout count not less than this threshold as high-fanout nets. You can also specify the threshold by using the **-threshold** option.

If you specify a value for *input_coll*, the command searches for high-fanout objects only from the specified collection, rather than from the entire design.

A standard Tcl collection is returned. All of the generic collection manipulating commands, such as **foreach_in_collection**, **sizeof_collection**, and so on, can be performed on this collection handle.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the collection of all high-fanout nets from the design:

```
prompt> all_high_transitive_fanout -nets
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

The following example shows the collection of all high-fanout nets from an existing collection stored in $COLL:

```
prompt> all_high_transitive_fanout -nets $COLL
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

The following example shows the collection of all high-fanout nets from the design having a fanout count of more than 100:

```
prompt> all_high_transitive_fanout -nets -threshold 100
{ii2/hin iv3/in iv3/out tin ii2/hout tout}
```

SEE ALSO

all_transitive_fanin(2)
all_transitive_fanout(2)
foreach_in_collection(2)
sizeof_collection(2)

all_inputs

Creates a collection of all input ports in the current design. You can assign these ports to a variable or pass them into another command.

SYNTAX

collection **all_inputs**
[-level_sensitive]
[-exclude_clock_ports]
[-edge_triggered]
[-clock *clock_name*]

Data Types

clock_name list

ARGUMENTS

-level_sensitive

Only considers ports with level-sensitive input delay. This is specified by **set_input_delay 2 -clock CLK -level_sensitive IN1**. This option cannot be used together with **-edge_triggered**.

-exclude_clock_ports

Excludes input ports which serve as clock sources.

-edge_triggered

Only considers ports with edge-triggered input delay. This is specified by **set_input_delay 2 -clock CLK IN2**. This option cannot be used together with **-level_sensitive**.

-clock *clock_name*

Only considers ports with input delay relative to a specific clock. This can be the name of a clock, or a collection containing a clock.

DESCRIPTION

The **all_inputs** command creates a collection of all input or inout ports in the current design. You can limit the contents of the collection by specifying the type of input delay that must be on a port.

You can remove clock source ports from the resulting collection by specifying *-exclude_clock_ports* option.

If you want only certain ports, use **get_ports** to create a collection of ports matching a specific pattern and optionally passing filter criteria.

When issued from the command prompt, **all_inputs** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable **collection_result_display_limit**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example specifies a driving cell for all input ports.

prompt> **set_driving_cell -lib_cell FFD3 -pin Q [all_inputs]**

SEE ALSO

collections(2)
get_ports(2)
query_objects(2)
report_ports(2)
set_driving_cell(2)
set_input_delay(2)
shell.common.collection_result_display_limit(3)

all_modes

Creates a collection of all modes in the current design. You can assign these modes to a variable or pass them into another command.

SYNTAX

collection **all_modes**
[-design *design_name*]

Data Types

design_name string

ARGUMENTS

-design *design_name*

Name of the top design. The default is the current design.

DESCRIPTION

The **all_modes** command creates a collection of all modes in the current design. If you do not define any modes, a collection containing the default mode is returned.

If you want only certain modes, use **get_modes** to create a collection of modes matching a specific pattern and optionally pass in filter criteria.

Multicorner-Multimode Support

This command works on all modes.

EXAMPLES

This script uses the **all_modes** command to perform **report_clocks** one time for each defined mode.

```
foreach_in_collection mode [all_modes] {
  current_mode $mode
  report_clocks
```

```
}
```

SEE ALSO

collections(2)
create_mode(2)
current_mode(2)
get_modes(2)
report_clocks(2)

all_outputs

Creates a collection of all output ports in the current design. You can assign these ports to a variable or pass them into another command.

SYNTAX

collection **all_outputs**
[-level_sensitive]
[-edge_triggered]
[-clock *clock_name*]

Data Types

clock_name list

ARGUMENTS

-level_sensitive

Only considers ports with level-sensitive output delay. This is specified by **set_output_delay 2 -clock CLK -level_sensitive IN1**. This option cannot be used together with **-edge_triggered**.

-edge_triggered

Only considers ports with edge-triggered output delay. This is specified by **set_output_delay 2 -clock CLK IN2**. This option cannot be used together with **-level_sensitive**.

-clock *clock_name*

Only considers ports with output delay relative to a specific clock. This can be the name of a clock, or a collection containing a clock.

DESCRIPTION

The **all_outputs** command creates a collection of all output or inout ports in the current design. You can limit the contents of the collection by specifying the type of output delay that must be on a port.

If you want only certain ports, use **get_ports** to create a collection of ports matching a specific pattern and optionally passing filter criteria.

When issued from the command prompt, **all_outputs** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable

collection_result_display_limit.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example specifies a pin capacitance for all output ports.

```
prompt> set_load 4.56 [all_outputs]
```

The following example shows the names all of the output ports with output delay relative to PHI1.

```
prompt> all_outputs -clock PHI1
```

SEE ALSO

collections(2)
get_ports(2)
report_ports(2)
query_objects(2)
set_output_delay(2)
shell.common.collection_result_display_limit(3)

all_registers

Creates a collection of register cells or pins. You can assign the resulting collection to a variable or pass it into another command.

SYNTAX

collection **all_registers**
[-clock *clock_name*]
[-rise_clock *rise_clock_name*]
[-fall_clock *fall_clock_name*]
[-cells]
[-data_pins]
[-clock_pins]
[-async_pins]
[-output_pins]
[-level_sensitive]
[-edge_triggered]

Data Types

clock_name list
rise_clock_name list
fall_clock_name list

ARGUMENTS

-clock *clock_name*

Considers all registers clocked by *clock_name*. This is either the name of a clock or a collection containing a clock. For example, all registers whose clock pins are in the fan out of the specified clock.

-rise_clock *rise_clock_name*

Considers all registers clocked by *rise_clock_name* and having open edge effectively the rising clock edge. This is either the name of a clock, or a collection containing a clock. For example, rising edge triggered flip-flops without any inversion on the clock path or falling edge triggered flip-flops with inversion on the clock path.

-fall_clock *fall_clock_name*

Considers all registers clocked by *fall_clock_name* and having open edge effectively falling the clock edge. This is either the name of a clock, or a collection containing a clock. For example, rising edge triggered flip-flops with inversion on the clock path or falling edge triggered flip-flops without any inversion on the clock path.

-cells

Creates a collection of cells (default). The cells are registers and are further limited by other command options.

-data_pins

Creates a collection of register data pins. The collection can be limited by other command options.

-clock_pins

Creates a collection of register clock pins. The collection can be limited by other command options.

-async_pins

Creates a collection of asynchronous preset or clear pins.

-output_pins

Creates a collection of register output pins.

-level_sensitive

Limits search to level-sensitive latches.

-edge_triggered

Limits search to edge-triggered flip-flops.

DESCRIPTION

The **all_registers** command creates a collection of pins or cells related to registers.

When issued from the command prompt, **all_registers** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the variable **collection_result_display_limit**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following command returns all registers in the current design.

```
prompt> all_registers
```

SEE ALSO

collections(2)
get_cells(2)
get_pins(2)

shell.common.collection_result_display_limit(3)

all_scenarios

Creates a collection of all scenarios in the current design.

SYNTAX

collection **all_scenarios**
[-design *design*]

Data Types

design collection

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

DESCRIPTION

The **all_scenarios** command creates a collection of all scenarios in the current design, or the design specified by the **-design** option.

If you want only certain scenarios, use **get_scenarios** to create a collection of scenarios matching a specific pattern and optionally pass in filter criteria.

Multicorner-Multimode Support

This command works on all scenarios.

SEE ALSO

collections(2)
report_scenarios(2)
create_scenario(2)
current_scenario(2)
get_scenarios(2)

all_transitive_fanin

Reports pins, ports, or cells in the fanin of specified sinks.

SYNTAX

list **all_transitive_fanin**
-to *sink_list*
[-startpoints_only]
[-only_cells]
[-flat]
[-levels *count*]

Data Types

sink_list list
count int

ARGUMENTS

-to *sink_list*

Reports a list of sink pins, ports, or nets in the design and connectivity based fanin of each sink in the *sink_list*. If you specify a net, the effect is the same as listing all driver pins on the net.

-startpoints_only

Returns only the connectivity startpoints.

-only_cells

Results in a set of all the cells in the connectivity fanin of the *sink_list*.

-flat

Specifies to function in the flat mode of operation. The two major modes in which **all_transitive_fanin** functions are hierarchical (default) and flat. When in hierarchical mode, only objects from the same hierarchy level as the current sink are returned. Thus, pins within a level of hierarchy lower than that of the sink are used for traversal but they will not be reported.

-levels *count*

Stops traversal when reaching the perimeter of the search of *count* hops, where counting is performed over the layers of cells that are of equidistant from the sink.

DESCRIPTION

This command reports the connectivity fanin of specified sink pins, ports, or nets in the design. A pin is considered to be in the connectivity fanin of a sink if there is a path through combinational logic from the pin to that sink. The fanin report stops at the clock pins of registers (sequential cells).

NOTE: This command reports results except the cases

1. *false paths* do not have an impact on this commands ability to trace logic

2. manually created clocks are not treated as paths start points

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show the connectivity fanin of a port in the design. The design comprises three inverters in a chain named *iv1*, *iv2*, and *iv3*. The *iv1* and *iv2* inverters are hierarchically combined in a larger cell named *ii2*.

```
prompt> all_transitive_fanin -to tout
{"ii2/hin", "iv3/in", "iv3/out", "tin", "ii2/hout", "tout"}

prompt>  all_transitive_fanin -to tout -flat

{"ii2/iv1/U1/a", "ii2/iv2/U1/z", "tin", "iv3/U1/a", "ii2/iv1/U1/z",
"ii2/iv2/U1/a", "iv3/U1/z", "tout"}
```

SEE ALSO

all_transitive_fanout(2)
all_fanin(2)
all_fanout(2)
report_transitive_fanin(2)

all_transitive_fanout

Returns a set of pins, ports, or cells in the fanout of the specified sources.

SYNTAX

list **all_transitive_fanout**
-from *source_list*
[-endpoints_only]
[-only_cells]
[-flat]
[-levels *count*]

Data Types

source_list list
count int

ARGUMENTS

-from *source_list*

Specifies a list of source pins, ports, or nets in the design. The connectivity fanout of each source in *source_list* is reported. If a net is specified, the effect is the same as listing all load pins on the net.

-endpoints_only

Returns only connectivity endpoints as a result.

-only_cells

Results in a set of all cells in the connectivity fanout of the *source_list*, rather than a set of pins or ports.

-flat

Specifies to function in the flat mode of operation. The two major modes in which **all_transitive_fanout** functions are hierarchical (default) and flat. When in hierarchical mode, only objects from the same hierarchy level as the current source are returned. Thus, pins within a level of hierarchy lower than that of the source are used for traversal but are not reported.

-levels *count*

Stops traversal when reaching the perimeter of the search of *count* hops, where counting is performed over the layers of cells that are of equidistant from the source.

DESCRIPTION

This command reports the connectivity fanout of specified source pins, ports, or nets in the design. A pin is considered to be in the connectivity fanout of a sink if there is a path through combinational logic from that source to the pin. The fanout report stops at the inputs to registers (sequential cells).

NOTE: This command reports results except the cases

1.**false paths** do not have an impact on this commands ability to trace logic 2.Manually created clocks are not treated as paths start points

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the connectivity fanout of a port in the design. The design comprises three inverters in a chain named *iv1*, *iv2*, and *iv3*. The *iv1* and *iv2* inverters are hierarchically combined in a larger cell named *ii2*.

```
prompt> all_transitive_fanout -from tin
{"iv3/out", "tout", "iv3/in", "ii2/hin", "ii2/hout", "tin"}

prompt> all_transitive_fanout -from tin -flat
{"tout", "ii2/iv2/U1/z", "ii2/iv1/U1/a", "iv3/U1/z", "iv3/U1/a",
"ii2/iv2/U1/a", "ii2/iv1/U1/z", "tin"}

prompt> all_transitive_fanout -from tin -levels 1 -only_cells
{"iv3", "ii2"}
```

SEE ALSO

all_fanout(2)
all_fanin(2)
all_transitive_fanin(2)
report_transitive_fanout(2)

analyze

Analyzes the specified HDL source files and stores the resulting templates into the specified library in a format ready to specialize and elaborate to form linkable cells of a full design.

SYNTAX

status **analyze**
[-format verilog | sverilog | vhdl]
[-standard 1987 | 1993 | 1995 | 2001 | 2005 | 2008 | 2009 | 2012]
[-define *define_list*]
[-hdl_library *hdl_library_name*]
[-uses *design_libs_list*]
[-vcs *vcs_opts*]
[-update]
[-recursive]
[-autoread]
[-rebuild]
[-output_script *output_string*]
[-exclude *exclude_list*]
[-verbose]
[-top *top_design_name*]
file_list

Data Types

file_list list
hdl_library_name string
design_libs_list string
define_list string
vcs_opts string
output_string string
top_design_name string
exclude_list list

ARGUMENTS

-hdl_library *hdl_library_name*

Specifies the template library where analyzed output files are stored. When the -hdl_library option is not used, the hdl_library_name is taken from **hdlin.hdl_library.default_name**. If the directory for the hdl_library_name is not defined, then **define_hdl_library** is automatically called to do so.

-uses *design_libs_list*

This option is not available with the **-format vhdl** or **-autoread** option.

SystemVerilog and Verilog designs might contain unresolved references to unelaborated design templates without indication of where those template definitions are located.

The compiler and linker always search for these templates first in the home directory of the parent design. By default, they then continue each template search by visiting every design library defined by the **define_hdl_library** command. The **-uses** option overrides this last step. It stores a design library search order within each design being analyzed. Later, elaboration or linking will follow this prescribed order rather than visiting every known library. An empty list argument limits the search to just the home library of the parent design. You do not need to repeat the **-hdl_library** setting, and you cannot override a search of the work directory.

-standard 1987 | 1993 | 1995 | 2001 | 2005 | 2008 | 2009 | 2012

Specifies the revision year of the IEEE standard that governs the selected language's syntax and semantics. The default revision years are IEEE Std 1076-2008 for VHDL, IEEE Std 1800-2012 for SystemVerilog, and IEEE Std 1364-2005 for Verilog.

-format verilog | sverilog | vhdl

Specifies the format of the files to be analyzed. The supported formats are Verilog, SystemVerilog, and VHDL.

Specifies that the **-autoread** option should look for files in the specified language whose extensions match the extensions in the list identified by the **hdlin.autoread.verilog_extensions**, **hdlin.autoread.sverilog_extensions**, or **hdlin.autoread.vhdl_extensions** application option. The **-autoread** option supports Verilog, SystemVerilog, and VHDL formats. By default, **-autoread** reads in all the files it finds that have Verilog, SystemVerilog, and VHDL extensions. You can use the **-autoread** option to specify file names in the *file_list* that do not have one of the language-specific extensions.

-vcs *vcs_opts*

Specifies all VCS-specific command-line options. The options must be specified as a single string, such as enclosed within braces ({}) or double quotation marks (""). Options specified by the **-vcs** option follow the VCS command-line syntax:

[-sverilog | -verilog] Specifies the format of the files to be analyzed.

[-y *directory_path*] Specifies the directories that contain the library files to be searched for unresolved module instantiations in the design.

[+libext+.*extension1*+...] Specifies the extension to consider during a file search in the -y library directories. The default is no extension.

[-v *library_file*] Holds several module definitions to be used during the search for unresolved modules.

[-f *command_file*] Specifies a command file.

[+define+*macro_name*+...] Defines a macro.

[+incdir+*dir1*+...] Includes directories in the search list.

src_file1 Specifies the files that are to be analyzed.

This option is not available with the **-autoread** option.

-define *defines_list*

Specifies a list of constant macros to be defined for the session of the **analyze** command. This is useful for the inclusion of Verilog or SystemVerilog code enclosed by `ifdef/`endif constructs. For details, see the documentation.

file_list

Specifies the files to be analyzed.

To analyze multiple files, list them by using curly braces, {}. Note: Source files in a list are effectively concatenated into a single

source stream in the listed order.

In **-autoread** mode only, the *file_list* argument can also specify directory names. All HDL source files located in those directories are prescanned. For more information about prescanning, see the descriptions for the **-autoread** and **-recursive** options.

-autoread

Specifies to use the **-autoread** mode for script-free analysis of whole HDL source directories or directory trees. All HDL source files located in those directories are prescanned, grouped, and then ordered into distinct source code streams based on their mutual inclusion and package-reference dependences. Some or all source streams are finally analyzed using the basic command.

By default, the **-autoread** option creates or replaces only a subset of analyzed result files; it works in an incremental-update mode that resembles the Unix make command. If an HDL source file does not exist or is newer than the intermediate files it should produce, only that file and the source streams that require it are reanalyzed.

For more information, see the description for the **-rebuild** option.

-top *design_name*

Identifies the top-level component of a hierarchical design specified within the **-autoread** HDL source directories.

The **analyze -autoread -top** *design name* command selects a subset of the indicated HDL source streams to analyze. Its goal is to analyze into intermediate form precisely those source files required for a later elaboration and linking of *design_name*. Files in the indicated directories that are not required to elaborate the design hierarchy descending from *design_name* are prescanned but the design templates they would produce are not created or replaced.

If the **-top** option is not provided, all HDL source streams indicated by the **file_list** argument are analyzed.

This option can be used only with the **-autoread** option.

-recursive

Specifies that the **analyze -autoread** command must recursively scan the directory trees below those specified in the *file_list* list. By default, the **-autoread** option looks only in the specified directories.

This option can be used only with the **-autoread** option.

-exclude *exclude_list*

Specifies a list of files and directories that are not to be analyzed. An **analyze -autoread [...] file_list** command checks each HDL source path against all elements of the *exclude_list*; it ignores a file if it or any of its directory paths matches any excluded path.

This option can be used only with the **-autoread** option.

-rebuild

Replaces any result files regardless of their timestamp. It analyzes all the indicated HDL source streams as if the result library was empty.

This option can be used only with the **-autoread** option.

-verbose

Prints out more messages for the **-autoread** option.

This option can be used only with the **-autoread** option.

-output_script *file_name*

Creates a Tcl script that is compatible with Design Compiler, Fusion Compiler, and Formality.

The file reading order for third-party tools and a DOT language file dependency graph are embedded in the script as comments.

This option is available only with the **-autoread** option.

DESCRIPTION

This command translates the specified HDL files and stores the intermediate format in the specified hdl_library.

The following paragraphs describe the support for the **-autoread** option:

The **-autoread** option reads in the files from file_list, determines the dependencies between them, and analyzes them in the right order to prevent errors due to missed or misplaced files.

The dependencies are calculated only from the files or directories present in *file_list*. If *file_list* changes between consecutive calls with the **-autoread** option, the dependency inference is performed over the latest set of files provided. Based on this, all required sources must be present in *file_list* option or be available if the **-recursive** option is used on each call with the **-autoread** option.

If the top design name is provided by the **-top** option, only the source files required for elaborating that top design are analyzed. This filtering is based on file dependencies previously inferred. If the **-top** option is not provided, all sources are ordered, grouped, and analyzed per the dependency inferred between them.

To locate the source files, the command expands each item in the *file_list* list with the **search_path** variable. Then, it determines whether the result is excluded by the **-exclude** option or the **hdlin.autoread.exclude_extensions** application option. If it is not excluded and a file ends with one of the extensions in the **hdlin.autoread.vhdl_extensions** application option list, it is considered a VHDL source file. If the file ends with one of the extensions in the **hdlin.autoread.verilog_extensions** application option list, it is considered a Verilog source file. If the file ends with one of the extensions in the **hdlin.autoread.sverilog_extensions** application option list, it is considered a SystemVerilog source file.

When expanding a directory, the command collects the files from the directory, and from its subdirectories and theirs recursively if the **-recursive** option is set. The command then performs all extension checks on these files. If the **-format** option is set, only files with Verilog, SystemVerilog, or VHDL extensions are collected based on the value of the option.

After the **-autoread** option collects all of the source files, it performs the following dependency checks:

- Detects analyze dependencies: List RTL files in the right order for analyzing. For example, analyze the file that contains a VHDL entity before analyzing files that defines architectures of that entity, or analyze the file that contains a SystemVerilog package declaration before the files that import that package.
- Detects Verilog and SystemVerilog compilation unit dependencies: Determine if a file needs to be analyzed in the same compilation unit with other files. For example, if a file defines macros, SystemVerilog local parameter, and SystemVerilog enumerated values, and the file is not explicitly included by the file that requires that definitions, the **-autoread** option groups them in the same compilation unit in the correct order. This might not always be possible, such as when a macro is defined several times in different files and the **-autoread** option cannot determine which of those alternatives is the right choice.
- Detects link dependencies: Schedule the analyze stage for files required for elaboration of the design hierarchy. For example, if a Verilog or SystemVerilog design that is instantiated in one source file is declared on a different source file provided in the *file_list*, the second file also requires to be analyzed for a complete top-down design elaboration.
- Detects include dependencies: If a Verilog or SystemVerilog file is included in another source and the file changes between two consecutive calls of the **-autoread** option (in the same synthesis session), the **analyze** command includes this file and all files that include them to update the design.
- Infers the target library for VHDL files. However, if the **-library** option is specified, this step is skipped and VHDL files provided in the *file_list* are analyzed into the specified design library.

After the dependency check, all the required HDL files specified by the *file_list* option (regarding its dependencies and the **-top**

design option if defined) are analyzed.

When the **-autoread** option is used again (with the same *file_list* setting) after a file is changed, only the updated source files are analyzed.

The **-autoread** option executes the dependency analysis based only on the current *file_list* information. All HDL source files that might have relevant dependency relationships should be passed together in one **analyze -autoread** command.

EXAMPLES

The following example analyzes the packages.vhd file and stores the results in the MY_LIB HDL library:

```
prompt> analyze -format vhdl -hdl_library my_lib packages.vhd
```

The following example analyzes two SystemVerilog clients of two libraries:

```
prompt> define_hdl_library BUSDEFN -path /remote/IP/metra/big_bus
prompt> define_hdl_library NClib   -path ../../NCD/libNC
prompt> analyze -format sverilog -uses { BUSDEFN  NClib }  { senderAlice.sv receiverBob.sv }
```

The following example assumes that the current directory is the source directory. It specifies the source file list early in the command line and gives options, including the name of the top-level entity later.

```
prompt> analyze {.} -autoread -recursive -top E1
```

The next example specifies extensions for Verilog files that are different from the default ({.v}), sets the source list and the exclude list, and calls the command with the name of the top-level module name, forcing it to only collect files with Verilog extensions:

```
prompt> set_app_options -name hdlin.autoread.sverilog_extensions -value {.ve .VE}
prompt> set my_sources {mod1/src mod2/src}
prompt> set my_excludes {mod1/src/incl mod2/src/incl}
prompt> analyze $my_sources -exclude $my_excludes -autoread \
  -format verilog
```

Note that excluding include directories explicitly is only necessary if include files have the same extensions as source files and not all include files are included in the source.

SEE ALSO

define_hdl_library(2)
elaborate(2)
report_hdl_libraries(2)
set_top_module(2)
hdlin.hdl_library.default_name(3)

analyze_datapath_extraction

Analyzes DesignWare datapath extraction.

SYNTAX

status **analyze_datapath_extraction**
[-html_file file]
[-max_msgs *max_num_msgs*]
[-no_autoungroup]
[-no_report]
[-nosplit]
[module_list]

ARGUMENTS

-html_file *file*

Redirect the output of the report to specified files in HTML format.

-max_msgs *max_num_msgs*

Specifies the maximum number of messages of each type to display in each module.

-no_autoungroup

Specifies that automatic ungrouping is disabled. All hierarchies are preserved unless otherwise specified. The **-no_autoungroup** option in **analyze_datapath_extraction** behaves the same way as the **-no_autoungroup** option in **compile**. If you plan to specify the **-no_autoungroup** option in **compile**, you should also specify the **-no_autoungroup** option in **analyze_datapath_extraction**.

-no_report

Specifies not to report contained resources at the end of the analysis.

-nosplit

Does not split lines if column overflows.

module_list

Specifies list of modules to run datapath extraction analysis. If it's not specified, the whole current design will be analyzed.

DESCRIPTION

This command analyzes the datapath extraction of the current design.

When a DesignWare operator is not extracted, it analyzes the reason why it is not extracted. If it is due to the RTL coding, the tool issues a message about the RTL coding style that interferes with datapath extraction.

After the analysis, a report shows the contained resources in the design. This report helps to find the resources in the design source code.

The messages are sorted based on the priority. From high to low, the message types are "HDL-120", "HDL_132", "HDL-129", "HDL-133", "HDL-122", "HDL-125", "HDL-126", and "HDL-121". Within the same type, the messages are also sorted based on their potential impact.

For big designs, there could be a large number of messages. To make the output more readable, besides the **-max_msgs** option, the **set_msg** command can also be used in advance to control the desired number of each message in total to show, or just filter out some of the messages.

For more information about RTL coding styles that help improve datapath extraction, see the coding guidelines documented in "Coding Guidelines for Datapath Synthesis" on SolvNet (https://solvnet.synopsys.com/retrieve/015771.html).

Here are examples of possible warning messages issued by the **analyze_datapath_extraction** command:

"HDL-120": Issued when a DesignWare operator is breaking the datapath due to leakage on its fanout or the fanout of its driver. This DesignWare operator can be the boundary node of an extracted datapath block.

Datapath leakage occurs when an internal operand is not wide enough to store the result of an operation but the full result is required later. Operands with leakage must be binary and must be at the boundary of a datapath block. Therefore, operands with leakage will break a potentially larger datapath block into smaller datapath blocks, resulting in suboptimal QoR.

Example 1: module test(a, b, c, d, e, f, o1, o2, o3); input [10:0] a, b; input [5:0] c, d, e, f; output [8:0] o1; output [9:0] o2; output [7:0] o3; assign o1 = a + b + c; (add_9, add_9_2) assign o2 = f + o1; (add_10) assign o3 = e + o1; (add_11) endmodule

In this design, the input 'o1' of 'add_10' is truncated. Therefore, the following message is issued:

Information: Operator associated with resources 'add_10' in design 'test' breaks the datapath extraction because there is leakage due to truncation on the fanout of its driver 'add_9 add_9_2'. (HDL-120)

On the other hand, the width of 'o3' is smaller than the width of 'o1'. Therefore, there is no leakage on the input of 'add_11".

Example 2: module leakage_a (a, b, c, z); input signed [7:0] a, b; input [7:0] c; output [9:0] z; wire signed [8:0] t; assign t = a + b; //add_6 // leakage: signed result used in wider unsigned expression assign z = t + c; endmodule

There is leakage due to sign mismatch at the fanin(signed) and fanout(unsigned) of add_6:

Information: Operator associated with resources 'add_6' in design 'leakage_a' breaks the datapath extraction because there is leakage due to driver/load sign mismatch. (HDL-120)

Example 3: module leakage_b (a, b, c, z); input [7:0] a, b; input signed [7:0] c; output signed [10:0] z; wire signed [8:0] t;

assign t = a + b; // add_7 // leakage: unsigned result used in wider signed expression assign z = t + c; endmodule

There is leakage due to sign mismatch at the fanin(unsigned) and fanout(signed) of add_7:

Information: Operator associated with resources 'add_7' in design 'leakage_b' breaks the datapath extraction because there is leakage due to driver/load sign mismatch. (HDL-120)

Subtraction can give negative results that cannot be represented with an unsigned operand of limited width. It is recommended to declare the operands as signed. Note that constants used in signed operations need to be marked signed as well.

Example 4: It is recommended to change the following RTL code: input [13:0] a ; input [13:0] b ; wire [15:0] o = a - 14'd4 - b ;

to

input signed [13:0] a ; input signed [13:0] b ; wire signed [15:0] o = a - 14'sd4 - b;

The "HDL-120" message is associated with coding guideline 19. (https://solvnet.synopsys.com/retrieve/015771.html#leakage).

"HDL-121": There is a sequential cell that breaks the datapath extraction.

Example: module test (clk, en, a, b, c, z); input clk, en; input [7:0] a, b; input [15:0] c; output [15:0] z; reg [15:0] prod;

always @ (posedge clk) begin if (en) begin prod <= a * b; end end assign z = prod + c; endmodule

There is a register between the multiplier and the adder. The following message will be issued:

Information: There is sequential cell in between operator associated with 'mult_11' and 'add_15' in design 'test'. (HDL-121)

The extracted datapath block cannot contain sequential cells (for example, registers). The register in between operators will break a potential larger datapath block into smaller datapath blocks, resulting in suboptimal QoR.

The "HDL-121" message is associated with coding guideline 12. (https://solvnet.synopsys.com/retrieve/015771.html#pipelining).

"HDL-125": Design contains instantiated DW. If the instantiated DW is replaced with an inferred DW OP, it could be extracted.

Example: module test(a,b,c,z); parameter wl = 5; input [wl-1:0] a,b,c; output [2*wl-1:0] z;

wire [wl-1:0] sum; assign sum = (b+a); DW02_mult #(wl, wl)U_mult(sum, c, 1'b0, z); endmodule

The instantiated DW02_mult can be replaced by an inferred multiplier. The following message will be issued:

Information: Cell 'U_mult' in design 'test' can not be extracted because it instantiates 'DW02_mult', which could be inferred with operator '*' instead. (HDL-125)

The instantiated DesignWare component cannot be extracted into datapath blocks. If the RTL design uses inferred the DesignWare operation instead of instantiated DesignWare components, the inferred DesignWare operator might be extracted into a datapath block.

The "HDL-125" message is associated with coding guideline 11. (https://solvnet.synopsys.com/retrieve/015771.html#component_instantiation).

EXAMPLES

The following example shows the analysis report that the datapath extraction is broken due to a number of reasons.

```
prompt > analyze_datapath_extraction -no_report

****************************************
Datapath Extraction Analysis Messages
****************************************

Information: Operator associated with resources 'add_23 (new_02.v:23)  add_23_2 (new_02.v:23) ' in design 'dp_block_s' breaks t
Information: Operator associated with resources 'add_74 (new_02.v:74) ' in design 'test' breaks the datapath extraction because the
Information: Operator associated with resources 'mult_58 (new_02.v:58) ' in design 'test' breaks the datapath extraction because th
Information: Operator associated with resources 'add_24 (new_02.v:24) ' in design 'dp_block_s' breaks the datapath extraction bec
Information: Operator associated with resources 'add_28 (new_02.v:28)  add_28_2 (new_02.v:28)  add_28_3 (new_02.v:28) ' in de
Information: The output of subtractor associated with resources 'sub_72 (new_02.v:72) ' is treated as signed signal. (HDL-132)
Information: The output of subtractor associated with resources 'sub_25 (new_02.v:25) ' is treated as signed signal. (HDL-132)
Information: Cell 'multp_U (new_02.v:61) ' in design 'test' cannot be extracted because it instantiates 'DW02_multp', which could be
Information: Missing possible extraction across cell 'add_68 (new_02.v:68)' in design 'test' and cell 'add_23 (new_02.v:23)' in desig
Information: Missing possible extraction across cell 'add_68 (new_02.v:68)' in design 'test' and cell 'mult_24 (new_02.v:24)' in desig
```

```
Information: Missing possible extraction across cell 'add_59 (new_02.v:59)' in design 'test' and cell 'add_23 (new_02.v:23)' in desig
Information: Missing possible extraction across cell 'add_59 (new_02.v:59)' in design 'test' and cell 'mult_24 (new_02.v:24)' in desig
Information: Missing possible extraction across cell 'add_59 (new_02.v:59)' in design 'test' and cell 'sub_25 (new_02.v:25)' in desig

****************************************
Analyze Datapath Extraction Summary
****************************************

 Design     HDL-120  HDL-125  HDL-126  HDL-132  Total
------------------------------------------------------
 test           2        1        5        1      9
 dp_block_s     3        0        0        1      4
------------------------------------------------------
 Total          5        1        5        2     13

1
```

SEE ALSO

report_resources(2)

analyze_datapath_library_cell

Displays function, area and power characteristics of cells from the technology libraries used in datapath generation.

SYNTAX

status **analyze_datapath_library_cell**
[-html_file file]

ARGUMENTS

-html_file

Redirect the output of the report to specified files in HTML format.

DESCRIPTION

This command displays function, area and power characteristics of cells of the library that are used in datapath generation. It also checks for cells or combinations of cells that can improve power QoR during datapath generation.

EXAMPLES

The following examples use the demo class.nlib library. The **analyze_datapath_library_cell** command prints some general attributes of the library.

```
-----------------
 Library(s) Used
-----------------

  class (File: /u/test/class.nlib)
  Library units: time=1.00ns voltage=1.00V capacitance=1.00pF dynamic_power=1.00pW leakage_power=1.00pW
```

This description is followed by four tables. The first table lists significant cells used in datapath generation.

```
----------------------------------------
 Cells Selected for Datapath Generation
----------------------------------------

                          Best Dynamic  Best Leakage
```

```
 Cell Type  Fastest     Smallest   Power       Power
-----------------------------------------------------------------
 addab      ADDHX8M     ADDHX1M    ADDHX1M     ADDHX1M
 addabc     ADDFHX8M    ADDFHXLM   ADDFHXLM    ADDFHXLM
 addabcd    *NA*        *NA*       *NA*        *NA*
 addnabc    *NA*        *NA*       *NA*        *NA*
 and2       CLKAND2X16M AND2X2M    AND2X2M     AND2X2M
 aoi21      AOI21X8M    AOI21X1M   AOI21X1M    AOI21X1M
 aoi22      AOI22X4M    AOI22X1M   AOI22X1M    AOI22X1M
 aoi222     AOI222X2M   AOI222XLM  AOI222XLM   AOI222XLM
 benc       *NA*        *NA*       *NA*        *NA*
 bmux       *NA*        *NA*       *NA*        *NA*
 brec       *NA*        *NA*       *NA*        *NA*
 brecen     *NA*        *NA*       *NA*        *NA*
 bsel       *NA*        *NA*       *NA*        *NA*
 inverter   CLKINVX40M   INVX2M     INVX2M      INVX2M
 maj23      *NA*        *NA*       *NA*        *NA*
 maj23n     *NA*        *NA*       *NA*        *NA*
 mux2       MX2X12M     MX2X1M     MX2X1M      MX2X1M
 mux4       MX4X8M      MX4X2M     MX4X2M      MX4X2M
 muxi2      MXI2X12M    MXI2X2M    MXI2X2M     MXI2X2M
 nand2      NAND2X6M    NAND2XLM   NAND2XLM    NAND2XLM
 oai21      OAI21X2M    OAI21X1M   OAI21X1M    OAI21X1M
 oai22      OAI22X8M    OAI22X1M   OAI22X1M    OAI22X1M
 oai222     OAI222X4M   OAI222X1M  OAI222X1M   OAI222X1M
 ordemux    *NA*        *NA*       *NA*        *NA*
 xnor2      XNOR2X8M    XNOR2X2M   XNOR2X2M    XNOR2X2M
 xnor3      XNOR3X8M    XNOR3XLM   XNOR3XLM    XNOR3XLM
 xor2       CLKXOR2X16M XOR2X2M    XOR2X2M     XOR2X2M
 xor3       XOR3X8M     XOR3XLM    XOR3XLM     XOR3XLM
```

The second table shows specific attributes of the cells listed in the first table.

```
------------------------------
 Library Cell Characteristics
------------------------------

                Cell   Cell     Dynamic     Leakage
 Library Cell   Area   Drive    Power       Power
 ------------------------------------------------------------
 ADDFHX8M       86.2400  *NA*  3.5573e-03  0.0000e+00
 ADDFHXLM       32.4800  *NA*  8.9730e-04  0.0000e+00
 ADDFX8M        39.2000  *NA*  1.7905e-03  0.0000e+00
 ADDHX1M        15.6800  *NA*  4.3033e-04  0.0000e+00
 ADDHX8M        49.2800  *NA*  2.2714e-03  0.0000e+00
 AND2X2M         5.6000  *NA*  2.7210e-04  0.0000e+00
 AOI21X1M        5.6000  *NA*  1.2111e-04  0.0000e+00
 AOI21X6M       14.5600  *NA*  5.4362e-04  0.0000e+00
 AOI21X8M       19.0400  *NA*  7.0137e-04  0.0000e+00
 AOI222X2M      11.2000  *NA*  3.7821e-04  0.0000e+00
 AOI222X4M      20.1600  *NA*  7.5592e-04  0.0000e+00
 AOI222XLM      10.0800  *NA*  2.0234e-04  0.0000e+00
 AOI22X1M        7.8400  *NA*  1.7501e-04  0.0000e+00
 AOI22X4M       13.4400  *NA*  5.3030e-04  0.0000e+00
 CLKAND2X16M    20.1600  *NA*  1.5817e-03  0.0000e+00
 CLKINVX40M     30.2400  *NA*  1.8411e-03  0.0000e+00
 CLKMX2X16M     22.4000  *NA*  1.4179e-03  0.0000e+00
```

```
CLKXOR2X16M  52.6400  *NA*  3.2150e-03  0.0000e+00
INVX2M       3.3600   *NA*  1.0640e-04  0.0000e+00
INVX32M      24.6400  *NA*  1.5502e-03  0.0000e+00
<truncated>
```

The third table lists characteristics for the lowest area and power versions of the library cells. The area, power, and drive-strength characteristics are checked for potential problems and get listed in the fourth table.

The cell characterization table contains the following information for each cell type:

```
Smallest Cell    : The smallest cell that is found in the library
Total # Drives   : Number of different drive strengths for this cell
Min Area # Drives : Number of different drive strengths for the minimum area version of this cell
Estim Size       : Estimated size of the cell based on the estimated grid size
Actual Size      : Cell size from the library data
Dynamic Power    : Dynamic power from the library data
Min # Trans      : Estimated minimum number of transistors required to implement the cell
Estim # Grids    : Estimated number of grids required to implement the cell
Actual # Grids   : Calculated number of grids for the cell based on the area of the smallest inverter in the library
```

When a potential problem in the cell is detected, an asterisk (*) is added to the value in the characterization table.

```
-------------------------------------------------
 Cell Characterization for Lowest Area / Power
-------------------------------------------------

  Cell     Smallest    Total  Min Area  Estimated   Actual  Dynamic   Min  Estimated  Actual
  Type     Cell        #Drives  #Drives     Size      Size    Power  Trans    #Grids  #Grids
 ---------------------------------------------------------------------------------------------
  addab    ADDHX1M       4       1*   13.4400   15.6800   0.0004    14       8      10*
  addabc   ADDFHXLM      7       3    25.2000   32.4800   0.0009    28       15     20*
  addabcd  *NA*        *NA*     *NA*     *NA*     *NA*    *NA*   *NA*     *NA*    *NA*
  addnabc  *NA*        *NA*     *NA*     *NA*     *NA*    *NA*   *NA*     *NA*    *NA*
  and2     AND2X2M      7       1*    6.7200    5.6000   0.0003     6       4       4
  aoi21    AOI21X1M     5       1*    6.7200    5.6000   0.0001     6       4       4
  aoi22    AOI22X1M     2*      1*    8.4000    7.8400   0.0002     8       5       5
  aoi222   AOI222XLM    3*      1*   11.7600   10.0800   0.0002    12       7       6
  benc     *NA*        *NA*     *NA*     *NA*     *NA*    *NA*   *NA*     *NA*    *NA*
  <truncated>

-------------
 Cell Issues
-------------

  Cell     Issue
 ------------------------------------------------------------------------------
  addab    no low power cell (should have at least 2 drives for minimum footprint)
           cell too large (too many grids)
  addabc   cell too large (too many grids)
  addabcd  Does not exist
  addnabc  Does not exist
  and2     no low power cell (should have at least 2 drives for minimum footprint)
  aoi21    no low power cell (should have at least 2 drives for minimum footprint)
  aoi22    not enough total drives (should have at least 4)
           no low power cell (should have at least 2 drives for minimum footprint)
  aoi222   not enough total drives (should have at least 4)
           no low power cell (should have at least 2 drives for minimum footprint)
```

<truncated>

SEE ALSO

report_power_library(2)

analyze_design_violations

Analyzes the specific violations in current design, identifies unfixable violations, and generates a summary report.

SYNTAX

status **analyze_design_violations**
[-type max_trans | static_noise | setup | hold | only_lib]
[-fanout *hfn_threshold*]
[-slack *small_violation_threshold*]
[-output *output_prefix*]
[-stage preroute | postroute]
[-drc_nets *nets*]
[-endpoints *pins*]
[-max_slack *slack_value*]
[-min_slack *slack_value*]

Data Types

hfn_threshold	integer
small_violation_threshold	float
output_prefix	string
nets	collection
pins	collection
slack_value	float

ARGUMENTS

-type max_trans | static_noise | setup | hold | only_lib

Specifies the type of violation to analyze. Only one type can be specified at each time. By default, the maximum transition violations (**max_trans**) are analyzed.

When type is **static_noise**, static noise violations are analyzed.

When type is **setup**, setup violations are analyzed. For each end point, only the worst path is picked up.

When type is **hold**, hold violations are analyzed. For each end point, only the worst path is picked up too.

When type is **only_lib**, only the library settings that affect optimization are checked. Some messages are reported when buffers or inverters are missing for entire design, for a particular voltage and for a particular site-row. Note that this checking also happens for other types.

-fanout *hfn_threshold*

Specifies the threshold which is used for high fanout net check. The default value at pre-route stage is the value from

opt.common.max_fanout, and the default value at post-route stage is the value from extrat.high_fanout_threshold.

-slack *small_violation_threshold*

Specifies the slack threshold in picoseconds which is used for checking of a small violation. A positive value is required. By default, any slack violation less than 5 ps (**-slack 0.005**) is considered a small violation.

-output *output_prefix*

Specifies the output file name. The command appends "$type.txt" to the prefix and overwrites the file if there has been one. The output contains a summary of analysis, followed by a list of detailed violations, printed one-by-one in descending order. Violations are also grouped by category. By default, the output file is analyze_design_violations.max_trans.txt.

-stage preroute | **postroute**

Specifies the routing stage used for analysis. Default is **preroute**. The fanout threshold is different at either stage.

-drc_nets *nets*

Specifies the nets with LDRC violation. Tool only performs analysis on these nets.

-endpoints *pins*

Specifies the endpoint pins with timing violation. Tool only performs analysis on these nets.

-max_slack *slack_value*

Specifies the maximum slack of violations that will be analyzed. The violations larger than the value are filtered. It can work with "-min_slack". When "-drc_nets" or "-endpoints" is used, this option is ignored.

-min_slack *slack_value*

Specifies the minimum slack of violations that will be analyzed. The violations larger than the value are filtered. It can work with "-max_slack". When "-drc_nets" or "-endpoints" is used, this option is ignored.

DESCRIPTION

This command analyzes the specified violations in current design and generates a summary report for them. The violations to be analyzed is specified by -type option. The analyzed result is summarized in log file and is elaborated in a standalone output file. It is important to understand the output.

The outputs for LDRC and timing are different. LDRC violations are net based and the result are categories of nets. Timing violations are path based and its result are paths grouped by timing constraints.

First, the output of LDRC is introduced, which is made by 7 parts.

1. Summary of total violations

   It presents the global view of violations, including violation count, worst and total slack.

2. Categories which causes or may cause LDRC violations unfixable.

   Categories of Reasons are listed one by one. The abbreviation of reason is in "()". Then the violation number, worst/total cost in the category and the percentage in total are listed. After the separator "|", it is pin info. If there is just "0" after the string of reason, it means no violation belongs to the category.

   Note that a violation can match different categories, but it will be put into one. There is internal priority of categories.

"Entire net is dont_touch(D)": All the net segments are dont_touch, no buffer can be added for fixing.

"Net between pad cell and port(PAD)": Such kind of nets are not touched by optimization.

"Tristate nets(TR)": Tristate nets are not touched by optimization.

"Nets with multiple drivers(MD)": Multi-driven nets are not touched by optimization.

"Nets in read-only sub blocks(RO)": Nets in read-only blocks are all dont_touch.

"Nets are highly blocked(BLK)": This means the net's bounding box are highly blocked (blockage ratio > 95%). The buffering will be very difficult, though detouring may solve it.

"Nets with port far from the core area(FP)": Port location is set far from core area. New buffer could not be added outside core area for fixing.

"Large input transition of driver(LI)": It checks the max_trans violation on the input side of driver cell. If there has been large max_trans violation (> 10 * *small_violation_threshold*) on them, current net is grouped here. It is used for max_trans analysis only.

3. Categories about the clock nets.

   Usually nets on clock network are not touched by signal nets optimization. But if they drive data pins, optimization should also make a fix.

   "Clock is used as clock only(COC)": Net is on clock network and drives only clock pins.

   "Clock is used as data path only(COD)": Net is on clock network and drives only data pins.

   "Clock is used as clock and data(CAD)": Net is on clock network and drives both clock pins and data pins.

   "Clock is used as others(CO)": Other definition about the clock nets.

4. Categories which help understanding of fixable violations.

   They are some properties which are not the unfixed reasons but give tips on what remaining violations are. Violations listed here are usually fixable.

   "Net is crossing high density area(DEN)": High-density can be a reason to drop a buffering solution. Here the density check is not so identical with optimization. It catches a net here when the density of net's bounding box is over 0.95.

   "Net is partially blocked(PB)": This category means there is high-density around any of the pins in the net.

   "Connect to excluded cell of XMB(XME)": The excluded cells of a module-based exclusive move bounds are allowed to place outside of the bounds. Once buffering happens, new buffer belongs to the bound and can only be placed inside bound. So, if excluded cells are far away from the bound, no buffering can fix the violation.

   "Connect to excluded cell of HMB(HME)": This is the same as "XME", the difference is that the type of move bound is module-based hard move bound. However, Hard move bounds are suggested to remove during optimization, since hard move bounds are not honored by optimization.

   "Nets across different exclusive MBs(DXM)": Violation on nets which are crossing boundaries of exclusive move bounds may be unfixed due to unknown limitation. User can pay attention to them.

   "Nets across different hard MBs(DHM)": Violation on nets which are crossing boundaries of exclusive move bounds may be unfixed due to unknown limitation. User can pay attention to them. However, Hard move bounds are suggested to remove during optimization, since hard move bounds are not honored by optimization.

   "Some segments of net are dont_touch(PD)": Some net segments of the net are dont_touch. Though there are still bufferable net segments, it is not sure whether buffering can happen.

   "High fanout nets(> $Threshold)(HF)": The nets whose fanout number is larger than $Threshold are grouped here. For pre-route

stage, the value comes from opt.common.max_fanout, and for post-route stage, the value comes from extrat.high_fanout_threshold. The threshold value is present here.

"Nets crossing different voltage areas(MV)": The nets crossing multiple voltage areas are quite complicated. They are grouped here but lack of sufficient analysis. User can refer to "check_bufferability" commands.

"Nets with small violation(> $Threshold)(S)": Small violations are not analyzed but it does not mean optimization does not work on them. The threshold value is listed here and can be changed by "-slack" option.

"Nets are too short(TS)": If the net length is smaller than a threshold, the net is grouped here. The threshold is decided by internal grid size. If a net has been short, buffering does not help for LDRC and setup. It is necessary to find other reason.

"Nets are long(> 400.0um)(LN)": The nets whose length is over 400 um are grouped here. It is not reasonable to leave so long nets. User needs to check out real reasons.

"Other violations(OV)": Those violations which could not be grouped to above categories are all here.

5. Distributions of fixable violations.

   For LDRC violations, there are four distributions, including slack, fanout, net length and constraint. The distributed violations are those fixable ones. Now the boundaries of distribution cannot be changed.

6. Top 10 Violations and their category given by abbreviation.

7. Guidance to find detailed report.

   The report file is $prefix.$type.txt, e.g. analyze_design_violations.max_trans.txt. In the report file, the info in the regular log is also printed. And following that, there is elaboration of violations. The elaboration is also grouped by category. This means violations with the same reason are shown together. In each category, the violations are printed in descending order. For each violation, there are net name, worst pin name, constraint, slack and scenario. For static noise violation, worst pin is always the net driver. This is an example of a violation.

```
ropt_net_329531
  Worst Pin: PMB_inst/rst_n                    0.200,  -0.006,  ta_test_ss28_0.75V_m0.60VS_0.60GS_m40C_SigC
```

For timing analysis, the output format is similar, but the context is quite different since timing is path-based. This is the introduction.

1. Summary of total violations

   This is similar to LDRC analysis. The summary includes violated path count, worst and total slack.

2. Categories listed by timing constraints that may be over-set.

   There are some descriptions with the word "large" in the categories. The reference is 10% of the clock period. "large" means that the checked thing is greater than the reference value. However, this does not mean the violation is unfixable due to the too "large" setting. It is just a potential reason. A path may match different categories, but it can only be grouped to the worst category. In the detailed log file, all categories are listed for top 10 paths.

   "Large clock skew(LCS)": clock skew is the gap between delay to clock pin of launch register and clock pin to capture register. For setup, it is clock delay of launch register minus clock delay of capture register. For hold, it is opposite. The larger clock skew is, the less room for either setup or hold fixing.

   "Large external input delay(LID)": External Input delay occupies arrive time. The larger it is, the less room for setup fixing.

   "Large driver adjustment(LDA)": Driver adjustment occupies arrive time. The larger it is, the less room for setup fixing.

   "Large clock uncertainty(LCU)": For setup, clock uncertainty tightens require time. The larger it is, the less room for setup fixing. For hold, clock uncertainty increases require time. The larger it is, the more delay at path should be added.

   "Large setup time of end point(LLS)": Setup time tightens require time. The larger it is, the less room for setup fixing.

   "Large external output delay(LOD)": For setup, external output delay tightens require time. The larger it is, the less room for setup

fixing. For hold, external output delay increases require time. The larger it is, the more delay at path should be added.

"Large hold time of end point(LLH)": Hold time increases require time of hold. The larger it is, the more delay on path should be added.

"Delay setting is illegal(ID)": Delay is set as negative value.

"Conflict with setup at endpoint(CSE)": This is for hold analysis only. When setup is also violated, fixing hold will degrade setup.

"Path ends up at scan in pin(SIP)": End point of path is scan in pin of register.

"Small violations(SM)": Small violations are not analyzed but it does not mean optimization does not work on them. The threshold is 5 ps and can be changed by "-slack" option.

"Remaining violating paths(OT)": It contains all violations which are not grouped to any of above categories.

3. Distribution of uncategorized violations by slack

   Distributed violations are from "Remaining violating paths". Now the boundaries of distribution cannot be changed.

4. Top 10 Violations and their category given by abbreviation.

5. Guidance to find detailed report.

   In the file of timing analysis report, it also prints the summary first and then the elaboration of paths. The elaboration is different from LDRC analysis. It does not follow the category. The paths are printed in the descending order of slack. Besides some timing info, the path analysis result is listed. There could be more than one category. For top 10 violations, the buffering capability of nets along the path is also analyzed, which uses the same way as max_trans analysis.

```
****************************************************************************
 # 1:EP-q0_srcc0_reg_27_/D, -0.215,(OT)
 Startpoint: u1_dpsp_macc/h1div_scale_f32_func_reg/CK
 Endpoint: q0_srcc0_reg_27_/D
 Scenario: setup_TT0P8V0C_typrc110c_FuncTT0P72V0C.WC
 Path Group: A_SCLK_DYN
 Slack: -0.215,  Require Time: 0.995, Num of Nets: 18
 Path analysis result: Remaining violating paths

 Nets analysis result:

 ****** Net analysis categories ******

* category  S12: Net is crossing high density area(DEN)       2 nets
   n6812
   vcc_out1

* category  S13: Net is partially blocked(PB)                5 nets
   n7021
   n367
   n10917
   u1_dpsp_macc/n483
   u1_dpsp_macc/h1div_scale_f32_func

* category  S21: Nets are too short(TS)                      9 nets
   N1405
   n7023
   n6809
   n6808
   n6800
```

```
    n7053
    ve_vcc_out1
    u1_dpsp_macc/n722
    u1_dpsp_macc/n720

  * category  S23: Other violations(OV)                    2 nets
    n7018
    n9010
```

EXAMPLES

The following examples show different analysis:

prompt> **analyze_design_violations**

The command analyzes "max_trans" violations by default and generates the following report.

```
    Start analyzing max_trans violations.

    Detect 29 violating nets.
      Worst: -0.028     Total: -0.316     Average: -0.011
  1 violating nets are categorized.
      Worst: -0.013     Total: -0.013     Average: -0.013
  28 violating nets are uncategorized.
      Worst: -0.028     Total: -0.303     Average: -0.011

    Detect 252 violating pins.
      Worst: -0.028     Total: -2.622     Average: -0.010
  14 violating pins are categorized.
      Worst: -0.013     Total: -0.167     Average: -0.012
  238 violating pins are uncategorized.
      Worst: -0.028     Total: -2.455     Average: -0.010

    ****** Unfixable max_trans violations due to design and user constraints ******
  category                                  Net-cnt   Net-wst   Net-tot   Net-% | Pin-cnt   Pin-wst   Pin-tot   Pin-%
  ----------------                          -------   -------   -------   ----- | -------   -------   -------   -----
  * category  DRC1: Entire net is dont_touch(D)              0
  * category  DRC2: Net between pad cell and port(PAD)       0
  * category  DRC3: Tristate nets(TR)                        0
  * category  DRC4: Nets with multiple drivers(MD)           0
  * category  DRC5: Nets in read-only sub blocks(RO)         0
  * category  DRC6: Nets are highly blocked(BLK)             0
  * category  DRC7: Nets with port far from the core area(FP)    0
  * category  DRC8: Large input transition of driver(LI)     0

    ****** Clock network ******
  ----------------                          -------   -------   -------   ----- | -------   -------   -------   -----
  * category  DRC9: Clock is used as clock only(COC)         1    -0.013    -0.013    4.09% |    14    -0.013    -0.167    6.37%
  * category  DRC10: Clock is used as data path only(COD)    0
  * category  DRC11: Clock is used as clock and data(CAD)    0
  * category  DRC12: Clock is used as others(CO)             0

    ****** Remaining uncategorized fixable violations ******
```

```
----------------                                  -------  -------  -------  -----| -------  -------  -------  -----
* category  DRC13: Net is crossing high density area(DEN)        3   -0.027   -0.043  13.63% |    47   -0.027   -0.386  14.70%
* category  DRC14: Net is partially blocked(PB)                16   -0.028   -0.234  74.08% |   126   -0.028   -1.912  72.92%
* category  DRC15: Connect to excluded cell of XMB(XME)          0
* category  DRC16: Connect to excluded cell of HMB(HME)          0
* category  DRC17: Nets across different exclusive MBs(DXM)      0
* category  DRC18: Nets across different hard MBs(DHM)          0
* category  DRC19: Some segments of net are dont_touch(PD)       0
* category  DRC20: High fanout nets(> 40)(HF)                   0
* category  DRC21: Nets crossing different voltage areas(MV)     0
* category  DRC22: Nets with small violation(> -0.005)(S)       9   -0.005   -0.026   8.20% |    65   -0.005   -0.158   6.01%
* category  DRC23: Nets are too short(TS)                       0
* category  DRC24: Nets are long(> 400.0um)(LN)                 0
* category  DRC25: Other violations(OV)                         0

****** Remaining violations distributed by slack ******
-0.100 >= slack > -~                                0    0.000    0.000  -0.00% |    0    0.000    0.000  -0.00%
-0.050 >= slack > -0.100                            0    0.000    0.000  -0.00% |    0    0.000    0.000  -0.00%
-0.020 >= slack > -0.050                            3   -0.028   -0.081  25.76% |   26   -0.028   -0.661  25.19%
-0.010 >= slack > -0.020                           10   -0.019   -0.145  45.99% |   79   -0.019   -1.095  41.78%
-0.005 >= slack > -0.010                            6   -0.010   -0.050  15.96% |   68   -0.010   -0.541  20.65%
 0.000 >= slack > -0.005                            9   -0.005   -0.026   8.20% |   65   -0.005   -0.158   6.01%

****** Remaining violations distributed by fanout ******
  100 <= fanout < ~                                 0    0.000    0.000  -0.00% |    0    0.000    0.000  -0.00%
   40 <= fanout < 100                               0    0.000    0.000  -0.00% |    0    0.000    0.000  -0.00%
   20 <= fanout <  40                               2   -0.010   -0.012   3.74% |   58   -0.010   -0.313  11.94%
   10 <= fanout <  20                               4   -0.027   -0.039  12.35% |   45   -0.027   -0.509  19.40%
    5 <= fanout <  10                              16   -0.028   -0.207  65.74% |  127   -0.028   -1.584  60.41%
    0 <= fanout <   5                               6   -0.027   -0.044  14.08% |    8   -0.027   -0.049   1.88%

****** Remaining violations distributed by length ******
 800.0 <= length < ~                                0    0.000    0.000  -0.00% |    0    0.000    0.000  -0.00%
 400.0 <= length <  800.0                           0    0.000    0.000  -0.00% |    0    0.000    0.000  -0.00%
 200.0 <= length <  400.0                           0    0.000    0.000  -0.00% |    0    0.000    0.000  -0.00%
 100.0 <= length <  200.0                           7   -0.027   -0.080  25.25% |   61   -0.027   -0.809  30.84%
   0.0 <= length <  100.0                          21   -0.028   -0.223  70.66% |  177   -0.028   -1.646  62.79%

****** Remaining violations distributed by constraint ******
cstr = 0.277                                                  |  215   -0.028   -2.238  85.35%
cstr = 0.278                                                  |    8   -0.027   -0.088   3.36%
cstr = 0.281                                                  |    1   -0.015   -0.015   0.56%
cstr = 0.275                                                  |    9   -0.015   -0.078   2.96%
cstr = 0.300                                                  |    5   -0.027   -0.037   1.40%

Top 10 Violations:
# 1: aps_rename_37_, -0.028.(PB)
# 2: osrc_c_data[88], -0.027.(DEN)
# 3: n439, -0.027.(PB)
# 4: ovector_output[5], -0.019.(PB)
# 5: aps_rename_35_, -0.017.(PB)
# 6: aps_rename_41_, -0.016.(PB)
# 7: ovector_output[17], -0.016.(PB)
# 8: aps_rename_36_, -0.014.(PB)
# 9: ovector_output[4], -0.014.(PB)
#10: ovector_output[1], -0.013.(PB)
```

Please find detailed analysis in analyze_design_violations.max_trans.txt.

prompt> **analyze_design_violations -type static_noise**

The command analyzes "static_noise" violations and generates the following report.

```
Start analyzing max_peak_noise violations.

Detect 57 violating nets.
   Worst: -0.106      Total: -1.614      Average: -0.028
0 violating nets are categorized.
   Worst: 0.000      Total: 0.000      Average: 0.000
57 violating nets are uncategorized.
   Worst: -0.106      Total: -1.614      Average: -0.028

Detect 57 violating pins.
   Worst: -0.106      Total: -1.614      Average: -0.028
0 violating pins are categorized.
   Worst: 0.000      Total: 0.000      Average: 0.000
57 violating pins are uncategorized.
   Worst: -0.106      Total: -1.614      Average: -0.028

****** Unfixable max_peak_noise violations due to design and user constraints ******
category                                  Net-cnt    Net-wst    Net-tot    Net-% | Pin-cnt    Pin-wst    Pin-tot    Pin-%
-----------------                         -------    -------    -------    ----- | -------    -------    -------    -----
* category  DRC1: Entire net is dont_touch(D)                  0
* category  DRC2: Net between pad cell and port(PAD)           0
* category  DRC3: Tristate nets(TR)                           0
* category  DRC4: Nets with multiple drivers(MD)              0
* category  DRC5: Nets in read-only sub blocks(RO)            0
* category  DRC6: Nets are highly blocked(BLK)                0
* category  DRC7: Nets with port far from the core area(FP)   0
* category  DRC8: High fanout nets(> 1000)(HFR)               0

****** Clock network ******
-----------------                         -------    -------    -------    ----- | -------    -------    -------    -----
* category  DRC9: Clock is used as clock only(COC)            0
* category  DRC10: Clock is used as data path only(COD)       0
* category  DRC11: Clock is used as clock and data(CAD)       0
* category  DRC12: Clock is used as others(CO)                0

****** Remaining uncategorized fixable violations ******
-----------------                         -------    -------    -------    ----- | -------    -------    -------    -----
* category  DRC13: Net is crossing high density area(DEN)     0
* category  DRC14: Net is partially blocked(PB)               0
* category  DRC15: Connect to excluded cell of XMB(XME)       0
* category  DRC16: Connect to excluded cell of HMB(HME)       0
* category  DRC17: Nets across different exclusive MBs(DXM)   0
* category  DRC18: Nets across different hard MBs(DHM)        0
* category  DRC19: Some segments of net are dont_touch(PD)    0
* category  DRC20: Nets crossing different voltage areas(MV)  0
* category  DRC21: Nets with small violation(> -0.005)(S)     7    -0.005    -0.020    1.21% |     7    -0.005    -0.020    1.21%
* category  DRC22: Nets are too short(TS)                     0
* category  DRC23: Nets are long(> 400.0um)(LN)               0
* category  DRC24: Other violations(OV)                      50    -0.106    -1.594   98.79% |    50    -0.106    -1.594   98.79%
```

```
****** Remaining violations distributed by slack ******
-0.100 >= slack > -~                    1    -0.106    -0.106    6.57% |     1    -0.106    -0.106    6.57%
-0.050 >= slack > -0.100                9    -0.071    -0.547   33.93% |     9    -0.071    -0.547   33.93%
-0.020 >= slack > -0.050               24    -0.046    -0.748   46.38% |    24    -0.046    -0.748   46.38%
-0.010 >= slack > -0.020               13    -0.017    -0.166   10.27% |    13    -0.017    -0.166   10.27%
-0.005 >= slack > -0.010                3    -0.010    -0.026    1.64% |     3    -0.010    -0.026    1.64%
 0.000 >= slack > -0.005                7    -0.005    -0.020    1.21% |     7    -0.005    -0.020    1.21%

****** Remaining violations distributed by fanout ******
  100 <= fanout < ~                     0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
   40 <= fanout < 100                   0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
   20 <= fanout <  40                   0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
   10 <= fanout <  20                   0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
    5 <= fanout <  10                   1    -0.021    -0.021    1.30% |     1    -0.021    -0.021    1.30%
    0 <= fanout <   5                  56    -0.106    -1.593   98.70% |    56    -0.106    -1.593   98.70%

****** Remaining violations distributed by length ******
 800.0 <= length < ~                    0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
 400.0 <= length <  800.0               0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
 200.0 <= length <  400.0              16    -0.064    -0.505   31.32% |    16    -0.064    -0.505   31.32%
 100.0 <= length <  200.0              12    -0.106    -0.525   32.51% |    12    -0.106    -0.525   32.51%
  0.0 <= length <  100.0               29    -0.058    -0.584   36.17% |    29    -0.058    -0.584   36.17%

****** Remaining violations distributed by constraint ******
cstr = 0.308                                                    |    39    -0.071    -1.111   68.86%
cstr = 0.252                                                    |    18    -0.106    -0.502   31.14%

Top 10 Violations:
# 1: dc_dataram_wdata6[6], -0.106.(OV)
# 2: dc_dataram_wdata1[4], -0.071.(OV)
# 3: HFSNET_194, -0.068.(OV)
# 4: u_ca53_noram/u_ca53biu/u_ca53biu_data_write_buffers/ZINV_96, -0.064.(OV)
# 5: u_ca53_noram/u_ca53dpu/dftopt2551_gOb507, -0.064.(OV)
# 6: dc_dataram_wdata6[8], -0.063.(OV)
# 7: scu_dr_data_i[29], -0.058.(OV)
# 8: u_ca53_noram/ifu_cp_dbg_0[18], -0.055.(OV)
# 9: HFSNET_186, -0.053.(OV)
#10: u_ca53_noram/u_ca53dpu/FPU1_u_dpu_fp_u_dpu_fp_dp/fml1_b_data_f1[34], -0.050.(OV)

Please find detailed analysis in analyze_design_violations.max_peak_noise.txt.
```

prompt> **analyze_design_violations -type setup**

The command analyzes "setup" violations and generates the following report. Some path-based analysis summaries are shown in log.

```
Start analyzing SETUP violations.

Detect 5382 violating paths.
   Worst: -2.533      Total: -1992.886      Average: -0.370
266 violating paths are categorized.
   Worst: -1.307      Total: -114.315      Average: -0.430
5116 violating paths are uncategorized.
   Worst: -2.533      Total: -1878.571      Average: -0.367

****** Setup timing categories ******
category                                Count     Worst     Total   Percentage
```

```
----------------                              -------  -------  -------  -----
* category  S1: Large clock skew(LCS)                    0
* category  S2: Large external input delay(LID)        238   -1.276   -84.830   4.26%
* category  S3: Large driver adjustment(LDA)            0
* category  S4: Large clock uncertainty(LCU)            0
* category  S5: Large setup time of end point(LLS)      28   -1.307   -29.485   1.48%
* category  S6: Large external output delay(LOD)        0
* category  S7: Delay setting is illegal(ID)            0
* category  S8: Conflict with setup at endpoint(CSE)    0
* category  S9: Small violations(SM)                   45   -0.005    -0.127   0.01%
* category  S10: Remaining violating paths(OT)       5071   -2.533 -1878.444  94.26%

****** Remaining violations distributed by slack ******
-0.100 >= slack > -~                               4152   -2.533 -1830.973  91.88%
-0.050 >= slack > -0.100                            472   -0.100   -35.316   1.77%
-0.020 >= slack > -0.050                            289   -0.050   -10.200   0.51%
-0.010 >= slack > -0.020                             99   -0.020    -1.493   0.07%
-0.005 >= slack > -0.010                             59   -0.010    -0.463   0.02%

Top 10 violations:
# 1:EP-sv_avd_core_inst/DCF/OLDMA/sv_cpu_dma_ctl0/regrddat_reg_10_/d, -2.533,(OT)
# 2:EP-sv_avd_core_inst/DCF/OLDMA/sv_cpu_dma_ctl0/regrddat_reg_18_/d, -2.486,(OT)
# 3:EP-sv_avd_core_inst/DCF/OLDMA/sv_cpu_dma_ctl0/regrddat_reg_13_/d, -2.359,(OT)
# 4:EP-sv_avd_core_inst/DCF/OLDMA/sv_cpu_dma_ctl0/regrddat_reg_5_/d, -2.300,(OT)
# 5:EP-sv_avd_core_inst/DCF/OLDMA/sv_cpu_dma_ctl0/regrddat_reg_7_/d, -2.286,(OT)
# 6:EP-sv_avd_core_inst/DCF/OLDMA/sv_cpu_dma_ctl0/regrddat_reg_8_/d, -2.283,(OT)
# 7:EP-sv_avd_core_inst/DCF/OLDMA/sv_cpu_dma_ctl0/regrddat_reg_6_/d, -2.277,(OT)
# 8:EP-sv_avd_core_inst/DCF/OLDMA/sv_cpu_dma_ctl0/regrddat_reg_4_/d, -2.276,(OT)
# 9:EP-sv_avd_core_inst/DCF/OLDMA/sv_cpu_dma_ctl0/regrddat_reg_9_/d, -2.073,(OT)
#10:EP-sv_avd_core_inst/DCF/OLDMA/sv_cpu_dma_ctl0/regrddat_reg_25_/d, -2.016,(OT)

Please find detailed analysis in analyze_design_violations.setup.txt.
```

prompt> **analyze_design_violations -type hold**

The command analyzes "hold" violations and generates the following report. Some path-based analysis summaries are shown in log.

```
Start analyzing HOLD violations.

Detect 430 violating paths.
   Worst: -0.042     Total: -5.592     Average: -0.013
45 violating paths are categorized.
   Worst: -0.033     Total: -0.665     Average: -0.015
385 violating paths are uncategorized.
   Worst: -0.042     Total: -4.927     Average: -0.013

****** Hold timing categories ******
category                                  Count     Worst     Total   Percentage
----------------                          -------  -------  -------  -----
* category  H1: Large clock skew(LCS)          45   -0.033    -0.665   11.88%
* category  H2: Large clock uncertainty(LCU)    0
* category  H3: Large hold time of end point(LLH)    0
* category  H4: Large external output delay(LOD)    0
* category  H5: Delay setting is illegal(ID)        0
* category  H6: Conflict with setup at endpoint(CSE)    0
* category  H7: Path ends up at scan in pin(SIP)        0
```

```
* category  H8: Small violations(SM)                    95    -0.005    -0.215   3.84%
* category  H9: Remaining violating paths(OT)           290    -0.042    -4.712  84.28%

****** Remaining violations distributed by slack ******
-0.100 >= slack > -~                                    0     0.000     0.000  -0.00%
-0.050 >= slack > -0.100                                0     0.000     0.000  -0.00%
-0.020 >= slack > -0.050                               59    -0.042    -1.576  28.19%
-0.010 >= slack > -0.020                              184    -0.020    -2.785  49.81%
-0.005 >= slack > -0.010                               47    -0.010    -0.351   6.27%

Top 10 violations:
# 1:EP-q1_ve_clamp_nan_reg/EN, -0.042,(OT)
# 2:EP-q2_ve_clamp_reg/EN, -0.034,(OT)
# 3:EP-q3_ve_omod_reg_0_/EN, -0.034,(OT)
# 4:EP-trans64_lzc_q0_reg_4_/D, -0.033,(LCS)
# 5:EP-trans64_lzc_q0_reg_5_/D, -0.033,(LCS)
# 6:EP-q_ve_srcc_fb4_en2_reg/D, -0.032,(OT)
# 7:EP-q_ve_srcc_fb4_en1_reg/D, -0.032,(OT)
# 8:EP-q_ve_srca_fb4_en3_reg/D, -0.032,(OT)
# 9:EP-q3_ve_omod_reg_1_/EN, -0.032,(OT)
#10:EP-q_ve_srca_fb8_en3_reg/D, -0.032,(OT)

Please find detailed analysis in analyze_design_violations.hold.txt.
```

prompt> **analyze_design_violations -type max_trans -drc_nets {aps_rename_37_ osrc_c_data[88]}**

The command analyzes only "max_trans" violations on two nets and generates the following report.

```
Start analyzing max_trans violations.

Detect 2 violating nets.
   Worst: -0.028      Total: -0.055      Average: -0.027
0 violating nets are categorized.
   Worst: 0.000      Total: 0.000      Average: 0.000
2 violating nets are uncategorized.
   Worst: -0.028      Total: -0.055      Average: -0.027

Detect 10 violating pins.
   Worst: -0.028      Total: -0.274      Average: -0.027
0 violating pins are categorized.
   Worst: 0.000      Total: 0.000      Average: 0.000
10 violating pins are uncategorized.
   Worst: -0.028      Total: -0.274      Average: -0.027

****** Unfixable max_trans violations due to design and user constraints ******
category                                  Net-cnt    Net-wst    Net-tot    Net-% | Pin-cnt    Pin-wst    Pin-tot    Pin-%
-----------------                         -------    -------    -------    ----- | -------    -------    -------    -----
* category  DRC1: Entire net is dont_touch(D)                  0
* category  DRC2: Net between pad cell and port(PAD)           0
* category  DRC3: Tristate nets(TR)                           0
* category  DRC4: Nets with multiple drivers(MD)              0
* category  DRC5: Nets in read-only sub blocks(RO)            0
* category  DRC6: Nets are highly blocked(BLK)                0
* category  DRC7: Nets with port far from the core area(FP)   0
* category  DRC8: Large input transition of driver(LI)        0

****** Clock network ******
```

```
------------------                              -------  -------  -------  -----|-------  -------  -------  -----
* category  DRC9: Clock is used as clock only(COC)              0
* category  DRC10: Clock is used as data path only(COD)         0
* category  DRC11: Clock is used as clock and data(CAD)         0
* category  DRC12: Clock is used as others(CO)                  0

****** Remaining uncategorized fixable violations ******
------------------                              -------  -------  -------  -----|-------  -------  -------  -----
* category  DRC13: Net is crossing high density area(DEN)       1    -0.027   -0.027  49.44% |     1    -0.027   -0.027   9.86%
* category  DRC14: Net is partially blocked(PB)                1    -0.028   -0.028  50.56% |     9    -0.028   -0.247  90.14%
* category  DRC15: Connect to excluded cell of XMB(XME)         0
* category  DRC16: Connect to excluded cell of HMB(HME)         0
* category  DRC17: Nets across different exclusive MBs(DXM)     0
* category  DRC18: Nets across different hard MBs(DHM)          0
* category  DRC19: Some segments of net are dont_touch(PD)      0
* category  DRC20: High fanout nets(> 40)(HF)                   0
* category  DRC21: Nets crossing different voltage areas(MV)    0
* category  DRC22: Nets with small violation(> -0.005)(S)       0
* category  DRC23: Nets are too short(TS)                       0
* category  DRC24: Nets are long(> 400.0um)(LN)                 0
* category  DRC25: Other violations(OV)                         0

****** Remaining violations distributed by slack ******
-0.100 >= slack > -~                            0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
-0.050 >= slack > -0.100                        0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
-0.020 >= slack > -0.050                        2    -0.028    -0.055  100.00% |    10    -0.028    -0.274  100.00%

****** Remaining violations distributed by fanout ******
  100 <= fanout < ~                             0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
   40 <= fanout < 100                           0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
   20 <= fanout <  40                           0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
   10 <= fanout <  20                           0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
    5 <= fanout <  10                           2    -0.028    -0.055  100.00% |    10    -0.028    -0.274  100.00%

****** Remaining violations distributed by length ******
 800.0 <= length < ~                            0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
 400.0 <= length <  800.0                       0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
 200.0 <= length <  400.0                       0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
 100.0 <= length <  200.0                       0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
  0.0 <= length <  100.0                        2    -0.028    -0.055  100.00% |    10    -0.028    -0.274  100.00%

****** Remaining violations distributed by constraint ******
cstr = 0.277                                                |     8    -0.028    -0.220   80.34%
cstr = 0.278                                                |     1    -0.027    -0.027   9.80%
cstr = 0.300                                                |     1    -0.027    -0.027   9.86%

Top 10 Violations:
# 1: aps_rename_37_, -0.028.(PB)
# 2: osrc_c_data[88], -0.027.(DEN)

Please find detailed analysis in analyze_design_violations.max_trans.txt.
```

prompt> **analyze_design_violations -min_slack -0.01 -max_slack -0.005**

The command analyzes "max_trans" violations between -0.1ns and -0.05ns and generates the following report. The distribution of violations is shown at the beginning.

```
Start analyzing max_trans violations.

      < -0.010 :       14 -0.2
 -0.010 ~ -0.005 :       6 -0.1
 -0.005 ~ 0       :       9 -0.0

Detect 6 violating nets.
   Worst: -0.010      Total: -0.050      Average: -0.008
0 violating nets are categorized.
   Worst: 0.000      Total: 0.000      Average: 0.000
6 violating nets are uncategorized.
   Worst: -0.010      Total: -0.050      Average: -0.008

Detect 68 violating pins.
   Worst: -0.010      Total: -0.541      Average: -0.008
0 violating pins are categorized.
   Worst: 0.000      Total: 0.000      Average: 0.000
68 violating pins are uncategorized.
   Worst: -0.010      Total: -0.541      Average: -0.008

****** Unfixable max_trans violations due to design and user constraints ******
category                                  Net-cnt    Net-wst    Net-tot    Net-% | Pin-cnt    Pin-wst    Pin-tot    Pin-%
-----------------                         -------    -------    -------    ----- | -------    -------    -------    -----
* category  DRC1: Entire net is dont_touch(D)                  0
* category  DRC2: Net between pad cell and port(PAD)           0
* category  DRC3: Tristate nets(TR)                            0
* category  DRC4: Nets with multiple drivers(MD)               0
* category  DRC5: Nets in read-only sub blocks(RO)             0
* category  DRC6: Nets are highly blocked(BLK)                 0
* category  DRC7: Nets with port far from the core area(FP)    0
* category  DRC8: Large input transition of driver(LI)         0

****** Clock network ******
-----------------                         -------    -------    -------    ----- | -------    -------    -------    -----
* category  DRC9: Clock is used as clock only(COC)             0
* category  DRC10: Clock is used as data path only(COD)        0
* category  DRC11: Clock is used as clock and data(CAD)        0
* category  DRC12: Clock is used as others(CO)                 0

****** Remaining uncategorized fixable violations ******
-----------------                         -------    -------    -------    ----- | -------    -------    -------    -----
* category  DRC13: Net is crossing high density area(DEN)      2    -0.010    -0.016   31.69% |     46    -0.010    -0.359   66.21%
* category  DRC14: Net is partially blocked(PB)               4    -0.009    -0.034   68.31% |     22    -0.009    -0.183   33.79%
* category  DRC15: Connect to excluded cell of XMB(XME)        0
* category  DRC16: Connect to excluded cell of HMB(HME)        0
* category  DRC17: Nets across different exclusive MBs(DXM)    0
* category  DRC18: Nets across different hard MBs(DHM)         0
* category  DRC19: Some segments of net are dont_touch(PD)     0
* category  DRC20: High fanout nets(> 40)(HF)                  0
* category  DRC21: Nets crossing different voltage areas(MV)   0
* category  DRC22: Nets with small violation(> -0.005)(S)      0
* category  DRC23: Nets are too short(TS)                      0
* category  DRC24: Nets are long(> 400.0um)(LN)                0
* category  DRC25: Other violations(OV)                        0

****** Remaining violations distributed by slack ******
```

```
-0.100 >= slack > -~                    0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
-0.050 >= slack > -0.100                0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
-0.020 >= slack > -0.050                0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
-0.010 >= slack > -0.020                0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
-0.005 >= slack > -0.010                6    -0.010    -0.050  100.00% |    68    -0.010    -0.541  100.00%

****** Remaining violations distributed by fanout ******
  100 <= fanout < ~                     0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
   40 <= fanout < 100                   0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
   20 <= fanout <  40                   1    -0.010    -0.010   19.35% |    32    -0.010    -0.272   50.30%
   10 <= fanout <  20                   1    -0.006    -0.006   12.34% |    14    -0.006    -0.086   15.91%
    5 <= fanout <  10                   4    -0.009    -0.034   68.31% |    22    -0.009    -0.183   33.79%

****** Remaining violations distributed by length ******
 800.0 <= length < ~                    0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
 400.0 <= length <  800.0               0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
 200.0 <= length <  400.0               0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
 100.0 <= length <  200.0               0     0.000     0.000   -0.00% |     0     0.000     0.000   -0.00%
  0.0 <= length <  100.0                6    -0.010    -0.050  100.00% |    68    -0.010    -0.541  100.00%

****** Remaining violations distributed by constraint ******
cstr = 0.277                                                          |    65    -0.010    -0.528   97.51%
cstr = 0.275                                                          |     3    -0.007    -0.013    2.49%

Top 10 Violations:
# 1: n10203, -0.010.(DEN)
# 2: ovector_output[0], -0.009.(PB)
# 3: ovector_output[7], -0.009.(PB)
# 4: ovector_output[16], -0.009.(PB)
# 5: n7273, -0.007.(PB)
# 6: u3_dpsp_macc/n797, -0.006.(DEN)

Please find detailed analysis in analyze_design_violations.max_trans.txt.
```

SEE ALSO

report_constraints(2)
report_timing(2)
check_bufferability(2)

analyze_lib_cell_placement

Analyze a collection of library cells for their ability to be placed in the current design.

SYNTAX

status **analyze_lib_cell_placement**
-lib_cells *cells*
[-trials *num_trials*]
[-region {{*llx lly*} {*urx ury*} }]
[-no_pg_drc]
[-no-adv]
[-max_cells *rpt_max*]
[-threshold *pass_thresh*]
[-directory *drc__directory*]
[-drc_limit *num_drcs*]

Data Types

cells collection
num_trials integer
llx float
lly float
urx float
ury float
rpt_max integer
pass_thresh float
drc_directory string
num_drcs integer

ARGUMENTS

-lib_cells *cells*

Specifies the library cells to analyze for placeability. This is a required option.

-trials *num_trials*

Specifies the number of random sites at which to test the legality of each cell. The default is 1000. A value of 0 will force an exhaustive search of all free sites in the design.

-region { {*llx lly*} {*urx ury*} }

Specifies a bounding box in which to search for legal sites. Library cells will be considered legal for sites in the region only if they remain wholly withing the region.

-no_pg_drc

This option disables pnet checking (the "pg_drc" rule in the Advanced Legalizer).

-no-adv

No advanced rule checking will be performed if this option is specifed.

-max_cells *rpt_max*

Specifies the maximum number of cells to list in the report. The default is 100. The true limit is the minimum of *rpt_max* and the number of lib cells specified.

-threshold *pass_thresh*

Specifies the threshold pass rate. Only cells whose pass rate is less than this will be reported. The default is 1.0, that is, 100%.

-directory *drc__directory*

Specify a directory for DRC violation files. The default is the current directory.

-drc_limit *num_drcs*

Specify a limit for how many DRC violations are written to each DRC file. The default is 100.

DESCRIPTION

This command tests a collection of library cells for their ability to be placed in a the current design. For each library cell in the collection, *num_trials* free sites are selected at random. If a region is specified, then the list of random sites is limited to free sites within that area. If the cell can be placed at that site in any orientation, then it is considered to pass at that site. If it cannot, it is considered to fail at that site. The pass rate for the cell is the number of passing sites divided by the number of sites tested.

After all of the cells have been analyzed, a report is produced listing cells in order of increasing pass rate. The length of this report is limited by the *rpt_max* argument. It is also limited by the *pass_thresh* argument. No cells whose pass rate is above this will be reported.

With DRC enabled (without using the -no_pg_drc option), when a lib cell has DRC violations, a file is created containing the violations. The file is named *analyze.libCellName.drc*. By default these files are written to the current directory. You can specify a directory using the *-directory drc_directory* option. By default, up to 100 DRC violations will be written into the file. You can change the limit using the *-drc_limit num_drcs* option. Setting *num_drcs* to zero means to write all violations. Also note that whatever directory you choose, existing drc files will be overwritten.

Multi threading is available when using the advanced legalizer. The maximum number of threads is obtained from the value set with the **set_host_options** command, but may be lowered based on the size of the site map.

EXAMPLES

The following example analyzes all opened designs.

prompt> **analyze_lib_cell_placement -lib_cells [get_lib_cells -filter is_open==true]**

SEE ALSO

legalize_placement(2)
set_host_options(2)

analyze_mv_design

Analyzes multivoltage design connections

SYNTAX

status **analyze_mv_design**
-level_shifter
[-through]
[-global_report]
[-output <output_file>]
[-html]
[-csv]
[-nosplit]
[-verbose]
[-example]

Data Types

through list
output_file string

ARGUMENT

-level_shifter

Enables level-shifter analysis. Required option, must be used with either [-global_report] or with [-through].

-through

Specifies a net or pin, and will report the global net containing that pin/net.

-global_report

Make an analysis of level-shifter insertion on the whole design, and counts one error for each driver-load pair.

-priority

The errors for global_report are picked based on a priority list, with this command you can change the order in which an error is selected in a driver-load pair.

The available priorities are:

1.

Missing or Problem with UPF constraint (MPC).

An error will be classified as MPC if:

a.

There is no domain boundary

b.

There is a Missing strategy (LS or ISO)

c.

Strategy cannot be fulfilled because there is a problem that prevent LS insertion.

2.

Problems that result in the maximum number of violations for that global net (FO).

This counts the number of loads that a specific error is affecting, so the one that is affecting more loads is going to have more priority.

3.

Library violation (LV).

An error will be classified as LV if it is related to missing library cell, for example: problem with map command (LS or ISO), main rail violation or no library cell matches the operating conditions.

4.

Supply violation (SV).

An error will be classified as SV if there is any problem with supplies availability.

5.

Dont touch (DT).

Net has dont touch.

6.

Closer to load (CL) Last resource to avoid ties. This counts the number of power domain (PD) boundaries are between the error and the load, the one with less PD boundaries between the error and the load is going to be chosen if all other categories are the same.

The command will receive a list with the priority order, if this is not given the default priority will be used {MPC FO LV SV DT CL}

-verbose

Generates detailed information in the report. For -through, add the list of supplies and list of lib cells. And for -global_report add the list of drivers for each error.

-output <*output_file*>

Print report into <*output_file*>.

-html

Print report in HTML format.

-csv

Print report in CSV (Comma-Separated Value) format. *-output* option is required when using this option and cannot be used in conjuntion with the *-html* option

-nosplit

Most of the reported data is listed in fixed-width columns. If the data for a given column exceeds the column width, the next field begins on a new line, starting in the right column. The *-nosplit* option prevents this behavior, allowing all the data to be written on a single line.

-example

Show example of different usages for analyze_mv_design.

DESCRIPTION

The analyze_mv_design command performs multivoltage analysis and reports design details that can help understand why level-shifter insertion failed. You can perform level-shifter analysis by using the -level_shifter with -global_report or with -through. The level-shifter analysis through a global net lists variable settings, driver and load information, signal connections, library availability information, and errors and warning collected from create_mv_cells.

EXAMPLES

The following examples are for -level_shifter -global_report and -level_shifter -through:

```
prompt> analyze_mv_design -level_shifter -global_report
****************************************
Report : analyze_mv_design
Design : Top
Version: P-2019.03-SP2-BETA
Date   : Mon May 20 13:12:57 2019
****************************************
Quantity  Error    Description
-------------------------------------------------
188       MV-1102  Cannot insert enabled level shifter in the path from
                     driver pin '%s' (related supply net: %s [%s]) to load
                     pin '%s' (related supply net: %s [%s]) because there
                     is no isolation cell on the path.

-------------------------------------------------
2         MV-1114  Found level shifter strategy '%s' with -location self
                     on pin '%s' but tool cannot insert level shifter in
                     the power domain because net '%s' has a user don't
                     touch attribute.


prompt> analyze_mv_design -level_shifter -through MIDA/U0/Z
****************************************
Report : analyze_mv_design
Design : top
Version: P-2019.03-SP2-BETA
```

```
Date   : Mon May 20 13:16:31 2019
****************************************
--------------------------------------
 Non-default App. Option Info
--------------------------------------
 mv.upf.ignore_ls_main_rail    true
--------------------------------------
------------------------------------------------------------------------------
 Type      Name      Voltage         Power         Related      #Fanout
                       Range           Domain        Supplies
------------------------------------------------------------------------------
 Driver    MIDA/U0/Z  0.72-0.72       PD_MIDA       (VDD1, VSS)   0
 Load      MID/in     0.72-0.72       PD_TOP        (VDD1, VSS)   1
 Load      MIDB/in    1.08-1.08       PD_MIDB       (VDD2, VSS)   1
------------------------------------------------------------------------------
------------------------------------------------------------------------------
      PD Boundary       Constraint    PD
        Available Supplies     Warnings/Errors          Usable Lib. Cells
------------------------------------------------------------------------------
0     MIDA/U0/Z (Drive   -           PD_MIDA
      r) - MIDA/out (n
      et)
        3                  No Warnings/Errors Found          0
1     MIDA/out (net) -   LSA          PD_MIDA
      MIDA/out (Pin)
        3                  No Warnings/Errors Found          4
2.1   x_UPF_LS           Size only ce  PD_TOP
                    ll
        3                  No Warnings/Errors Found          4
3.1   ls_0 (net) - MID   Don't touch   PD_TOP
      B/in (Pin)         net
        3                  No Warnings/Errors Found          4
4.1   MIDB/in (Pin) -     -           PD_MIDB
      MIDB/in (net)
        3                  No Warnings/Errors Found          4
2.2   x (net) - MID/in   Don't touch   PD_TOP
       (Pin)             net, LS_MID
        3                   MV-1102 Cannot insert enabled le   4
                            vel shifter in the path from dri
                            ver pin 'MIDA/U0/Z' (related sup
                            ply net: VDD1 [0.72V]) to load p
                            in 'MID/BOT/U0/I' (related suppl
                            y net: VDD4 [1.08V]) because the
                            re is no isolation cell on the p
                            ath.
3.2   x (net) - MID/in   Don't touch   PD_TOP
       (Pin)             net, LS_MID
        3                  No Warnings/Errors Found          4
------------------------------------------------------------------------------
```

SEE ALSO

check_mv_design(2)

report_mv_cell(2)
report_mv_path(2)

analyze_power_plan

Analyzes a complete or partial power network for voltage (IR) drop and electromigration on the specified power and ground nets. This command also calculate track utilization by P/G in each layer.

SYNTAX

status **analyze_power_plan**
[-nets *net_list*]
[-power_budget *power_budget*]
[-voltage *voltage*]
[-read_power_file *power_file_name*]
[-analyze_power]
[-pad_references *power_pad_reference_name*]
[-use_terminals_as_pads]
[-report_track_utilization_only]

Data Types

net_list	collection or list
power_budget	float
voltage	float
power_file_name	string
power_pad_reference_name	string

ARGUMENTS

-nets *net_list*

Specifies the power or ground nets on which to perform power network analysis. This is a required option if the **-report_track_utilization_only** option is not specified.

-power_budget *power_budget*

Specifies a total power budget for the design on which to perform power network analysis. The power budget is divided among the instances according to the cell area. For hard macros or standard cells, the power budget is computed based on the percentage of the total area. For hierarchical blocks, the power is computed based on the total area of all their child cells and blocks inside. The unit is in mW.

For example, if the power budget for the entire design is 500 mW, the total area of the top-level blocks is 500,000 square microns, and one of the blocks has an area of 25,000 square microns, power network analysis allocates the power for that 25,000-square-micron block as follows:

500mW * (25,000/500,000) = 25mW

Either **-power_budget** or **-read_power_file** should be specified to provide power data.

-voltage *voltage*

Specifies the voltage of the net to be analyzed. The voltage is applied at the power pad. The voltage unit is volts. The default is 1.5.

-read_power_file *power_file_name*

Obtains the power calculation source from a side file. By default, this option is off. The supported power files are PT-PX power output file, and default power file. The format of the default power file is as follows. Instance *instance_name power_value*

-analyze_power

Specifies the option to get power consumption as report_power. The option will use it for IR-drop calculation.

-pad_references *pad_references*

Specifies the pad references and the associated net as follows

net_name:*pad_reference_name*

If net name is not specified, the pad reference is used for all the analyzed nets. By default, PNA will assume all the pads logically connected to signal **1** or **0** as power pads.

-use_terminals_as_pads

Use terminals of top block as power/ground pads. By default, the option is off. This option treats terminals as additional power/ground pads. They can exist while using virtual pads. PNA will not make virtual connection if no P/G wires connect to them.

-report_track_utilization_only

If the option is used, all the other options are not required. This option will report the percentage of track resource is occupied by P/G wires but skip calculating IR drop. If the option is not used, PNA will still calculate track utilization percentage and perform IR-drop and EM calculation.

DESCRIPTION

This command performs a power network analysis (PNA) in a complete or partial power/ground network during the design planning stage. If the power ports in standard cells and hard macros are not connected to the power and ground network, PNA creates virtual connections to those unconnected power ports. PNA then analyzes the IR drop and electromigration in the power network based on user-specified power budget and voltage supply. This command requires that at least one cell in the design is logically connected to the power or ground net being analyzed. If not, you should use the **connect_pg_net** command to create the logic connection first. If there are any power switches that are connected to the analyzed net, you should set up a suitable resistance to power switches before you use this command. If not, PNA will assign a very small resistance to all unspecified power switches. To set up resistance of power switches, you can use **set_attribute** to set power_switch_resistance or **plan.pna.power_switch_resistance** application option as follows:

set_app_options -name plan.pna.power_switch_resistance

This command also calculates track utilization. Power, ground and secondary nets are taken into account. Only preferred routing direction defined in tech file is considered as routing resource. The resource of each routing layer that P/G wires utilize is displayed in the log.

EXAMPLES

The following example analyzes the IR drop on the *VDD* net with a total power budget of *1000* mW.

```
prompt> analyze_power_plan -nets VDD -power_budget 1000
```

The following example analyzes the IR drop on the *VDD* net with a total power budget of *1000* mW using pad reference VDD_NS

```
prompt> analyze_power_plan -nets VDD -power_budget 1000 -pad_references "VDD_NS"
```

The following example analyzes the IR drop on the *VDD* net with a total power budget of *1000* mW using pad reference VDD_NS for VDD and VSS_NS for VSS.

```
prompt> analyze_power_plan -nets {VDD VSS} -power_budget 1000 \
  -pad_references "VDD:VDD_NS VSS:VSS_NS"
```

The following example analyze the IR drop on the *VDD* net with a total power budget of *1000* mW with power switch cell (50 Ohm)

```
prompt> set_attribute [get_cells power_switch_cell_name] power_switch_resistance 50
prompt> analyze_power_plan -nets {VDD} -power_budget 1000
```

or

```
prompt> set_app_options -name plan.pna.power_switch_resistance -value \
  {{power_switch_lib_cell_name1 50}}
prompt> analyze_power_plan -nets {VDD} -power_budget 1000
```

SEE ALSO

connect_pg_net(2)
set_virtual_pad(2)
plan.pna.power_switch_resistance(3)

analyze_rail

Performs the targeted rail analysis on the specified nets. This command generates the data needed to run the RedHawk tool and runs the RedHawk tool within the current tool session. This command enables several target analyses, such as power and ground network voltage drop analysis, electromigration analysis, and minimum path resistance analysis. A previously generated RedHawk command script can also be used to perform the analyses.

SYNTAX

status **analyze_rail**
[-redhawk_script_file *redhawk_script* | -nets *net_list*]
[-voltage_drop *static* | *dynamic* | *dynamic_vcd* | *dynamic_vectorless*]
[-switching_activity {*file_format* [*file_name*] [*strip_path*] [*time_window*]} | -power_analysis icc2]
[-electromigration]
[-min_path_resistance]
[-effective_resistance]
[-result_name *result_name*]
[-check_missing_via]
[-extra_gsr_option_file]
[-script_only]
[-submit_to_other_machines]
[-multiple_script_files {{*result_name script_file*}...}]
[-bg]
[-step_size *time*]
[-stop_time *time*]

Data Types

file_format	string
file_name	string
nets	collection or list
redhawk_script	string
result_name	string
strip_path	string
time	string

ARGUMENTS

-redhawk_script_file *redhawk_script*

Specifies and uses an existing RedHawk script to perform the RedHawk run.

This script is usually generated from an earlier **analyze_rail** run.

When a user-specified script file is used, all other currently enabled options are ignored. For example, if you run the **analyze_rail**

-voltage_drop static -redhawk_script_file myscript.tcl command, the -voltage_drop option is ignored.

This option and the -nets option are mutually exclusive.

-nets *net_list*

Specifies the power and ground supply nets to be analyzed. This option is required if the -redhawk_script_file option is not set.

The tool considers all the switched or internal power nets of the specified power nets in the analysis. You do not need to explicitly specify the internal power nets.

If the -nets option is selected, you need to specify at least one of the following options: *-min_path_resistance* or *-voltage_drop*.

This generates extra VDD_NETS and GND_NETS blocks in the GSR file.

Each specified power net must have a corresponding *supply_net* defined with its *voltage_min* attribute set. The *voltage_min* attribute may be set using the **set_voltage** command.

-voltage_drop *static* | *dynamic* | *dynamic_vcd* | *dynamic_vectorless*

Performs voltage drop analysis on the specified power and ground nets.

When this option is selected, the tool checks for voltage drop violations on the specified power and ground supply nets. The results are saved under the current rail database directory, set by the rail application option rail.database, or the RAIL_DATABASE directory (the default). After the /fBanalyze_rail/fP command completes operation, the **open_rail_result** command is automatically invoked so users can view results in the GUI. It generates a voltage drop map, a parasitics map, and a power map. In addition, it reports voltage drop violations in a text file and in an error view.

-switching_activity {*file_format file_name strip_path time_window*}

Specifies an optional switching activity file. The *file_format* argument can be *VCD* (a VCD file generated from a gate-level simulation), *FSDB*, *SAIF*, or *ICC_ACTIVITY*.

A VCD or SAIF file can reference a top-level design in a different hierarchy than the one used in the current session. The *strip_path* argument removes the specified string from the beginning of the object name. Using this option modifies the object names in the VCD or SAIF file to make them compatible with the object names in the design library.

For example, use the following option to load a VCD file and strip the /top_design/cup_module prefix from all object names.

```
-switching_activity {VCD top_design.vcd /top_design/cup_module}
```

For example, use the following option to load a VCD file without a strip path.

```
-switching_activity {VCD top_design.vcd}
```

If you have annotated switching activity on the design in the IC Compiler II environment, set the *file_format* argument to *ICC_ACTIVITY*. When you specify this format, the tool creates a temporary SAIF file that contains the user-annotated switching activity and passes it to the RedHawk tool. The *file_name* argument does not apply to this format.

When the *file_format* arguement is set to *VCD* or *SAIF*, the tool generates a VCD_FILE or SAIF_FILE block in the GSR file, respectively.

When the *strip_path* argument is specified, the tool generates the *FRONT_PATH* component in the VCD_FILE block, or the *ROOT_INSTANCE* component in the SAIF_FILE block. Both VCD_FILE and SAIF_FILE blocks are in the GSR file.

When the *time_window* argument is specified, the tool generates the *SELECT_RANGE* component in the VCD_FILE block in the GSR file.

-electromigration

Performs electromigration analysis on the specified power and ground nets.

When enabled, the tool analyzes the current density on the specified nets and identifies the segments with potential

electromigration issues. It also calculates current violations based on the default rules that are loaded in the design library. The results are saved under the current rail database directory, set by rail application option rail.database, or the RAIL_DATABASE directory (the default). After the **analyze_rail** command completes, the **open_rail_result** command is automatically invoked, so you can view the results in the GUI. It generates a electromigration map and reports the current violations in an error view.

The -electromigration option must to specified with the -voltage_drop option.

When analysis run is complete, the tool adds the RedHawk "perform emcheck" command in the run script.

-min_path_resistance

Performs minimum path resistance analysis on the specified power and ground nets. The results are saved under the current rail database directory, set by rail application option rail.database, or the RAIL_DATABASE directory (the default). After the **analyze_rail** command completes, the **open_rail_result** command is automatically invoked so you can view the results in the GUI. It generates a minimum path resistance map.

If you want to perform minimum path resistance analysis only, do not use the -voltage_drop option.

When analysis run is complete, the tool adds the RedHawk "perform gridcheck -limit -1 -fullchip" command to the run script.

-effective_resistance

Performs effective resistance analysis on the specified power and ground nets. The results are saved under the current rail database directory, set by rail application option rail.database, or the RAIL_DATABASE directory (the default). After the **analyze_rail** command completes, the **open_rail_result** command is automatically invoked so you can view the results in the GUI. It generates an effective resistance map.

When analysis run is complete, the tool adds the RedHawk "perform res_calc" command in the run script.

-power_analysis *icc2*

When *icc2* is selected, the native power analysis is used to generate necessary power data for rail analysis.

See the **report_power** command man page for more information.

-result_name *result_name*

Specifies the directory name for saving the rail analysis results. The results are saved in {design_name}/rail/{result_name} directory under the current rail database directory, set by the rail application option rail.database, or by default under the RAIL_DATABASE directory.

The default result name is REDHAWK_RESULT.

Subsequent analyses overwrite the previous analyses results.

This generates an INSTANCE_POWER_FILES block in the GSR file.

-check_missing_via

Performs missing via check.

When checking is complete, the tool adds the RedHawk "mesh vias -report_missing" command in the run script.

"-check_missing_via" requires "-voltage_drop".

-extra_gsr_option_file

Specifies an external GSR file to include at the tail of the generated GSR file.

This generates extra INCLUDE statements at the end of the GSR file, and overrides the application option settings.

-script_only

Generates the RedHawk script and the input data for the target analysis, such as voltage drop, minimum path resistance or electromigration, without actually executing the analysis run.

This allows you to modify and reuse the run script as desired.

-submit_to_other_machines

Launches RedHawk on a separate machine. This option should be used with the **set_host_options** command to set a specific machine or farm.

-multiple_script_files {{*result_name script_file*}...}

Launches multiple RedHawk jobs using multiple script files provided in the option. When analysis is finished, each result is saved with the result_name specified. This option has to be used with the -submit_to_other_machines option, and the **set_host_options** option where a farm machine system has to be set.

The number of licenses required is equal to the number of script files provided. If there are not enough licenses available, the **analyze_rail** command aborts with an error message. If you want to run **analyze_rail** when the number of available licenses is less than the number of scripts, use **set_app_options -name rail.license_checkout_mode -value wait** before invoking the **analyze_rail** command. The tool launches scripts based on the number of the available licenses, and puts other scripts waiting for an available license. A license is released from other sessions or from the same session when a script finishes the job. With this application option set to *wait*, the license checking is performed dynamically when scripts are running during the **analyze_rail** command run.

-bg

Launches RedHawk in the background process and returns shell to the user. Use the **report_background_jobs** command to check if a previous background job has completed. Below restrictions apply: - Recursive -bg is not allowed. - Overlapping background jobs is not allowed. - RAIL_DATABASE related commands and license-related commands are not allowed in the background jobs.

-step_size

For RedHawk-SC flow, generates the *step_size* argument used by **create_analysis_view** command in analyze_rail.py file generated by the **analyze_rail** command.

The default value: For technology 5nm or below, use 5ps as the step size. For technology above 5nm, use 10ps as the step size. If technology is not specified, by default technology is set to 7nm. Use the application option *rail.technology* to specify the technology.

-stop_time

For RedHawk-SC flow, generates the *max_duration* argument used by the **create_analysis_view** command in the analyze_rail.py file generated by the analyze_rail command.

The default value: Always use 4x dominant clock period as the maximum duration. If the calculated max duration is smaller than step size, then the maximum duration is equal to 10x step size.

DESCRIPTION

This command prepares the data needed for RedHawk rail analyses, and then calls RedHawk or RedHawk-SC to perform the specified rail analysis run. The supported rail analyses are voltage drop analysis, electromigration analysis, minimum path resistance analysis, and effective resistance analysis.

The results are saved under the current rail database directory, set by the rail application option rail.database, or the RAIL_DATABASE directory (the default). After the **analyze_rail** command completes, the **open_rail_result** command is

automatically invoked, so you can view the results in the GUI. The command generates error views that capture violations found during the analysis. You can display the generated errors in the error browser, depending on the analysis options you chose.

You can perform multiple analyses, such as voltage drop and electromigration, in a single run.

The generated analysis results include the following information:

- Average voltage drop values in a map file
- Average electromigration values in a map file
- An error cell with average electromigration values

There are several rail application options that are used by the **analyze_rail** command:

- rail.redhawk_path to specify the RedHawk executable path For example:

  ```
  prompt> set_app_options -name rail.redhawk_path \
    -value /release/latest/Testing/bin
  ```

- rail.database to specify where to save the RedHawk analysis results (the default is RAIL_DATABASE) For example:

  ```
  prompt> set_app_options -name rail.database -value my_rail_output
  ```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example performs static voltage drop analysis for the nets {VDD VDDV} using native power analysis:

```
prompt> create_taps -import my.tap
prompt> analyze_rail -nets {VDD VDDV} -voltage_drop static \
  -power_analysis icc2
```

The following example performs minimum path resistance analysis for the net VDD:

```
prompt> create_taps -import my.tap
prompt> analyze_rail -nets VDD -min_path_resistance
```

The following example specifies a time window for the VCD file which is used to run dynamic rail analysis on the net VDD.

```
prompt> analyze_rail -voltage_drop dynamic_vcd -nets vdd \
  -switching_activity {VCD top_design.vcd /top_design/cup_module \
    {start_time end_time}}
```

The following example specifies a time window for the FSDB file which is used to run dynamic rail analysis on the net VDD.

```
prompt> analyze_rail -voltage_drop dynamic_vcd -nets vdd \
  -switching_activity {FSDB top_design.fsdb /top_design/cup_module \
    {start_time end_time}}
```

The following example submits a RedHawk job to an LSF farm.

```
prompt> set_host_options -submit_protocol lsf -submit_command {bsub -P bnormal}
prompt> analyze_rail -voltage_drop dynamic -submit_to_other_machines
```

The following example launches multiple RedHawk jobs to farm machines to have both static and dynamic rail analyses running at the same time. Two SNPS_INDESIGN_RH_RAIL licenses will be checked out.

```
prompt> set_host_options -submit_protocol lsf -submit_command {bsub -P bnormal}
prompt> analyze_rail -multiple_script_files {{static static.tcl} {dynamic dynamic.tcl}} -submit_to_other_machines
```

The following example assumes only 2 licenses are available, but needs to launch 4 RedHawk script files in the same analysis run. Since **set_app_options -name rail.license_checkout_mode -value wait** is set, the **analyze_rail** command launches 2 scripts first and puts other 2 scripts waiting for a license to be available.

```
prompt> set_app_options -name rail.license_checkout_mode -value wait
prompt> set_host_options -submit_protocol lsf -submit_command {bsub -P bnormal}
prompt> analyze_rail -multiple_script_files {{result1 result1.tcl} {result2 resilt2.tcl} {result3 result3.tcl} {result4 result4.tcl}} -
```

The following example launches RedHawk in background process and shell is returned to user.

```
prompt> analyze_rail ... -bg
prompt>
```

SEE ALSO

create_taps(2)
close_rail_result(2)
get_taps(2)
open_rail_result(2)
set_host_options(2)
rail.database(3)
rail.redhawk_path(3)
rail.technology(3)
remove_taps(2)
report_background_jobs(2)
report_power(2)
report_rail_result(2)
report_rail_minimum_path(2)
report_taps(2)
write_sdc(2)
write_parasitics(2)

analyze_subcircuit

Invokes circuit simulation on a subcircuit of the current design. This command is suitable for analyzing clock mesh nets.

SYNTAX

status **analyze_subcircuit**
[-name *string*]
[-from *pins*]
[-to *pins*]
[-net *net*]
[-simulator *name_of_simulator*]
[-spice_header_files *list_of_files*]
[-driver_subckt_files *list_of_files*]
[-configuration *list_of_scenario_configurations*]
[-clock *clock_name*]
[-spef_input_file_suffix *string*]
[-rc_include_file_suffix *string*]
[-num_simulation_cycles *inf*]
[-extraction]
[-create_spice_deck]
[-run_simulation]
[-write_annotation]
[-apply_annotation]
[-include_power_analysis]
[-verbose]

Data Types

pins collection of pins
net collection of one net
list_of_files list
list_of_scenario_configurations list
clock_name string
name_of_simulator string

ARGUMENTS

-name *string*

Specifies the name of this subcircuit, which is used to distinguish it from other subcircuits in the same design. The string is used to construct both circuit elements and file names, so it must only contain alphanumeric characters and underscores. With that restriction, you can use any name, but if a clock is being analyzed you should use the name of the clock (if this is unique). Many files are generated based on this name. The name is also given to the output directory, to be created in the current working

directory. If no directory by this name exists, one is created. Because many files are created in this directory, you must specify a directory with write permissions and sufficient disk space.

-clock *clock_name*

Specifies for which clock domain the analysis should be done. This especially helps to select the appropriate clock path when multiple clocks arrive at pins of a cell in the clock tree.

-from *pins*

Specifies the startpoint or startpoints to analyze a circuit. This can be a single driving point of the clock tree, such as the clock source pin, or it can be a set of driver pins in the pre-mesh clock tree.

-to *pins*

Specifies a collection of load pins that are driven by the circuit. These define the endpoints of the clock tree paths to be analyzed.

-net *net*

Specifies a single net, whose load pins will be used instead of -to as the endpoints of the clock tree paths to be analyzed. Typically this would be a multidriver clock mesh net. Either -to or -net must be specified.

-simulator *name_of_simulator*

Specifies which SPICE simulator to use. Only finesim, hspice, and nanosim simulators are supported. The default is nanosim. The simulator you want to use must be available as a working and properly licensed Linux command in your shell path. You may need to adjust the path in your .cshrc or .login file.

-spice_header_files *list_of_files*

Specifies the locations of the SPICE device models of the transistors. SPICE device models are needed for all the transistors. You specify the device models with the file path, and the file paths are copied into the simulator input. If you specify multiple files, they are included in the order specified. Note that file locations are relative to the current working directory (not relative to the new working directory defined by -name.) Absolute file paths may also be used.

-driver_subckt_files *list_of_files*

Specifies the locations of the subcircuit models of the buffers in the clock mesh tree. The circuit simulator requires a subcircuit model for all the buffers in the clock mesh tree. You specify the subcircuit models with the file path, and the file paths are copied into the simulator input. If you specify multiple files, they are included in the order specified. Note that file locations are relative to the current working directory. Absolute file paths may also be used.

-configuration *list_of_scenario_configurations*

Customizes the SPICE settings for each scenario and max/min setting to be analyzed. The list can be of any length. Single curly braces are recommended. If Tcl variable substitution is desired, the list can be defined using "..." rather than {...}. Each -scenario_name that is specified is followed by the custom file settings for that scenario. Any or all of the four types of custom file settings may be supplied. For any combinations not specified, the general settings from the **-spice_header_files** and **-driver_subckt_files** options will be used.

The format of the *list_of_scenario_configurations* argument is:

```
-configuration {
  -scenario_name scenario
   [-max_driver_subckt_files max_files]
   [-max_spice_header_files header_max]
   [-min_driver_subckt_files min_files]
   [-min_spice_header_files header_min]
  [-scenario_name scenario]
   [-max_driver_subckt_files max_files]
   [-max_spice_header_files header_max]
```

```
    [-min_driver_subckt_files min_files]
    [-min_spice_header_files header_min]
    ...
}
```

-spef_input_file_suffix *string*

This option can be used to customize the flow for -create_spice_deck, by replacing the default SPEF parasitics file name with an alternate user-provided SPEF file. For example, this could be used if an external extraction tool is being called rather than native extraction and SPEF generation. The external SPEF file should be generated without a name mapping table: in the IC Compiler tool, use the **write_parasitics -no_name_mapping** command. For each scenario and max/min simulation, the custom SPEF file names should follow this pattern:

<subcircuit_name>.<scenario_name>.<max/min>.<suffix_name>

-rc_include_file_suffix *string*

This option can be used to customize the flow for -create_spice_deck, by replacing the default include file name, for the net parasitics RC data, with the given custom suffix. The default suffix is \'rc.sp\', so for example, \'rc_custom.sp\' could be specified here instead. Users running external extraction tools instead of -extraction can then generate a customized RC parasitics input file for SPICE, and include it in the top-level SPICE deck file with this option, instead of the default.

-num_simulation_cycles *int*

This option can be used to extend the SPICE simulation duration for the given number of clock periods. This is very useful for insuring that the simulated circuit voltages have stabilized after circuit startup. By default only a single clock period is simulated. A typical value might be 3.

-extraction

Controls whether the analysis performs native extraction of RC parasitics and writes its own SPEF files for use by the **-create_spice_deck** option. This is Step 1 of the 5-step analysis process. If none of the five flow-control options are specified, by default all 5 steps will be run. If any of those 5 options are specified, then only the named steps will run.

-create_spice_deck

Controls whether the analysis will create the top-level SPICE deck files whose names end with *.top.sp, *.rc.sp, and *.meas.cmd. The *.rc.sp file describes the net RC parasitics in native SPICE format, and is by default created here by reading in the SPEF files generated by the -extraction step. The *.top.sp file is the top-level SPICE command file. The *.meas.cmd file defines to SPICE how to measure arrival and transition times at the pins of the clock tree. This is Step 2 of the 5-step analysis process. If none of the five flow-control options are specified, by default all 5 steps will be run. If any of those 5 options are specified, then only the named steps will run.

-run_simulation

Controls whether the analysis will run the SPICE simulation for each active scenario and max/min setting. The -simulator option specifies which SPICE tool is to be used. The SPICE measurement output files from this step are inputs to the -write_annotation step. This is Step 3 of the 5-step analysis process. If none of the five flow-control options are specified, by default all 5 steps will be run. If any of those 5 options are specified, then only the named steps will run.

-write_annotation

Controls whether the analysis will create *.tcl timing annotation files for the clock tree, based on reading in the SPICE measurement output files from -run_simulation. These files define annotated transition time and arc delay for the analyzed portions of the clock tree. For multidriver clock mesh nets, they also apply set_disable_timing to disable all except one of those drivers. The annotated net arc delays are then defined from the one remaining driver only. This is Step 4 of the 5-step analysis process. If none of the five flow-control options are specified, by default all 5 steps will be run. If any of those 5 options are specified, then only the named steps will run.

-apply_annotation

Controls whether the analysis will source the *.tcl annotation files from -write_annotation, causing the set_disable_timing, set_annotated_delay, and set_annotated_transition constraints to be applied to the clock tree. This is Step 5 of the 5-step analysis process. If none of the five flow-control options are specified, by default all 5 steps will be run. If any of those 5 options are specified, then only the named steps will run.

-include_power_analysis

Annotates power measurement from spice simulation into the design. If this option is specified, it appends .meas statements to the *.meas.cmd files which measure the average power of each cell instance in the spice netlist. It also appends set_annotated_power command in the *.tcl annotation files from -write_annotation. These commands annotate the measured power back into the design.

-verbose

Displays the extraction and circuit simulation messages to the console. By default, these messages are redirected into log files in the output directory.

DESCRIPTION

In general, the tool performs timing prediction with static timing analysis. But for some purposes, such as clock meshes, it is necessary or desirable to use circuit simulation. This command generates the necessary files to run a SPICE circuit simulator, invokes the simulator, and reads the output of simulation back into your design as annotation constraints for delay and transition times. The annotation constraints are Tcl files which can be reused in later sessions, as long as the clock tree netlist is unchanged.

Files

Inputs required:

You must have a circuit-level model for each of the buffer gates in your clock tree, and a transistor model for each of the transistors in the circuit models.

Files generated:

This command creates a directory named using the -name string, if it is not already present, and populates it with files for simulation and annotation. These include the circuit files, command files, and measurement files. One of these files is a shell script that is used to invoke the circuit simulator. Simulation produces many output and log files, which also populate the directory. This command then processes those output files and generates an annotation file. This file can be sourced into your design, to update the delays and transition times of your subcircuit to match the circuit simulation results.

Limitations

The subcircuit to be is analyzed is modeled as being driven by a single clock waveform signal. That signal is applied at the pin or pins named by -from. A typical usage is to name the clock source pin using -from.

If you want to analyze a set of mesh drivers whose inputs are not yet connected, you can do this using the **-from** option by listing all the mesh driver input pins. For the purpose of simulation, these are all driven in parallel, with no skew among them. This type of simulation is only for early exploration, before the clock tree in the fanin of the mesh drivers has been designed.

Buffers that are "shorted together" in a mesh or other structure are supported. The resistance and capacitance of the mesh is determined by extraction, and is expressed in a SPEF parasitic data file. The SPEF file is generated natively by default, but manual use of external extraction tools such as StarRC is supported by the customized-flow options, which allow users to call an external tool with their own scripting, and then bring the resulting SPEF or other parasitic model files into the SPICE simulation flow.

The underlying circuit simulator is not case-sensitive, so elements "buff_27" and "BUFF_27" are indistinguishable, and the design

must not contain both names. For similar reasons, circuit elements should not contain the dot character ".".

The loads of the clock tree endpoints (flip-flops or latches) are modeled by lump capacitance, not by active transistor models, since the outputs of those gates are not part of the simulation. If this operation is done before the signal nets have been routed, which would normally be the case, then final gate placement, routing geometry and detailed placement might not match the geometry in the final chip. There is no modeling of IR drop in the power or ground wires. The actual VSS and VDD used for each buffer gate is assumed to be static and is determined before simulation starts.

For more information on simulation see the documentation for Hspice, Finesim, and Nanosim.

Multicorner-Multimode Support

This command works on all active scenarios.

EXAMPLES

In the following example, a clock tree called clk1 has been implemented with a set of specified library references to be used as buffers, driving a clock mesh net called mesh_net. This causes a new directory named clk1 to be created, containing the simulation and annotation files for the specified clock tree. The endpoints of the analyzed tree are the load pins of mesh_net.

After this command has been run, the **report_timing** command reflects the delays and transitions observed in the circuit simulation, if the -path_type full_clock_expanded reporting option is used to show the annotated arc delays and transitions in the clock tree. This example uses hspice as the simulator.

```
prompt> analyze_subcircuit -name clk1 \
  -from clk1_source_pin -net mesh_net \
  -simulator hspice \
  -spice_header_files original/models.sp \
  -driver_subckt_files {original/drivers.sp extra_cells.sp}
```

SEE ALSO

report_timing(2)
set_annotated_delay(2)
set_annotated_transition(2)
set_disable_timing(2)

analyze_timing_correlation

Analyzes timing correlation with the PrimeTime tool.

SYNTAX

status **analyze_timing_correlation**
-work_dir *dir_name*
-pt_exec_path *filename*
[-scenarios *scenarios_list*]
[-delay_calculation_style zero_interconnect | auto]
[-si_enable_analysis true | false]
[-waveform_analysis_mode disabled | full_design]
[-enable_ccs_rcv_cap true | false]
[-pba_mode *pba_mode*]
[-overwrite_work_dir]
[-clear_work_dir]
[-save_pt_session]
[-use_pt_save_session]
[-disable_compatibility_settings]
[-enable_compatibility_settings]
[-pass_rate_threshold *integer*]
[-pt_search_path *paths_list*]
[-script_only]
[-pt_pre_link_script *filename*]
[-pt_post_link_script *filename*]
[-pt_user_script *filename*]
[-verbose]
[-nworst integer]

Data Types

dir_name string
filename string
scenarios_list list
pba_mode string
paths_list list

ARGUMENTS

-work_dir *dir_name*

Specifies working directory used to store all intermediate files which could be generated by analyze_timing_correlation. The files includes the design-files like netlist/constraints/parasitics, per-scenario Tcl-scripts and timing-data.

-pt_exec_path *executable_path*

Specify the PrimeTime executable image used for correlation. For example, /global/apps/pt_2015.06/bin/pt_shell

-scenarios *scenarios_list*

Specify which scenarios to analyze. By default, all scenarios are analyzed.

-delay_calculation_style zero_interconnect | auto

Controls the type of delay calculation performed during timing correlation analysis. When not specified, the tool uses the value of the **time.delay_calculation_style** application option.

Valid values are **zero_interconnect** and **auto**. When specifying **zero_interconnect** the tool infers that both signal integrity and waveform analysis are to be disabled, unless explicitly specified.

-si_enable_analysis true | false

Controls whether signal integrity is considered during timing correlation analysis. When not specified, the tool uses the value of the **time.si_enable_analysis** application option. Setting this value to true implies that timing windows are enabled.

Valid values are **true** and **false**.

-waveform_analysis_mode disabled | full_design

When not specified, the tool uses the value of the **time.delay_calc_waveform_analysis_mode** application option.

Valid values are **disabled** and **full_design**.

-enable_ccs_rcv_cap true | false

Specifies whether or not the Composite Current Source (CCS) receiver capacitance model is enabled. When not specified the tool uses the value of the **time.enable_ccs_rcv_cap** application option.

Valid values are **true** and **false**.

-pba_mode pba_mode

Specifies whether to perform path-based analysis or graph-based analysis The default is to perform graph-based analysis.

Valid values are **none**, **path** and **exhaustive**.

-overwrite_work_dir

Initially overwrite the existing work_dir directory.

-clear_work_dir

Overwrite existing work_dir upon command completion.

-save_pt_session

Save PrimeTime-Sessions and store it under '-work_dir' for later use.

-use_pt_save_session

Reuse existing PrimeTime-save-session stored under '-work_dir'.

-disable_compatibility_settings

Disables correlation changes to timer settings.

When you use this option, the command uses the current tool timing context during correlation analysis. This is not recommended

for the best alignment with the PrimeTime tool.

-enable_compatibility_settings

Enables correlation changes to tool timer settings.

When you use this option, the command modifies the timing context during correlation analysis to provide the best alignment. This is recommended for the best alignment with the PrimeTime tool.

-pass_rate_threshold *integer*

threshold criteria for correlation measurement, the default is 3. By default, 0.5 is used when delay_calculation_style equals 'zero_interconnect'.

-pt_search_path *paths_list*

Additional search_path used to resolve linked-libraries in PrimeTime. For example, '-pt_search_path {path1 path2 path3...}'.

-script_only

Write the design collateral and setup Tcl script files only.

-pt_pre_link_script *script_path*

Tcl script file to be sourced before design has been linked in PrimeTime.

-pt_post_link_script *script_path*

Tcl script file to be sourced after design has been linked in PrimeTime.

-pt_user_script *script_path*

Specifies a user-defined PrimeTime script for correlation.

By default, the command generates the design files for the PrimeTime runs based on the current tool settings.

If you use this option, the command uses the specified PrimeTime script for the correlation report.

The command ignores this option if you use the **-pt_pre_link_script** or **-pt_post_link_script** option.

-verbose

Show more detail information.

-nworst integer

Specifies how many timing endpoints to report in verbose analysis. By default, 5 endpoints are reported.

DESCRIPTION

The **analyze_timing_correlation** command generates a correlation report between this tool and the PrimeTime tool.

By default, the command performs the following tasks:

1. Generates QoR-compatible design files for the PrimeTime tool based on the current tool settings for each scenario.
2. Invokes the PrimeTime tool to collect the PrimeTime timing data.
3. Collects the timing data from this tool.

4. Generates the correlation report.

Multicorner-Multimode Support

By default, this command applies to all scenarios.

EXAMPLES

The following example uses the PrimeTime image located at /bin/pt_shell to generate a correlation report. The intermediate data is stored in a subdirectory named tmp.

```
prompt> analyze_timing_correlation -work_dir tmp \
  -pt_exec_path /bin/pt_shell -overwrite_work_dir -save_pt_session
```

SEE ALSO

save_upf(2)
write_parasitics(2)
write_script(2)
write_verilog(2)

annotate_trace

Control annotation of traced command execution to the shell output.

SYNTAX

annotate_trace [-start|-stop]
[-profile profile_annotation_type]
[-log log_string]
[-quiet]

string *log_string*
string *profile_annotation_type*

ARGUMENTS

-start

This option is mutually exclusive with -stop. The option starts the annotation of traced command execution to the shell output. Each executed command string is annotated to the output prepended with ";## ".

-stop

This option is mutually exclusive with -start. This option stops annotation of traced command execution to the shell output.

-profile *profile_annotation_type*

This option selects the profile data output behavior for command annotation. This option is only meaningful when given with the -start option. The argument is one of: on, with_start, off. The /flon/fP value starts annotation with profile data output enabled in the default mode which is to output the data only at the end of a command invocation. The data is output only if the delta values from the beginning of the command invocation meet or exceed the thresholds set for each of the reported metrics. The /flwith_start/fP value starts annotation with profile data output both at the beginning and at the end of a command invocation. The profile data is output at the beginning irregardless of any thresholds since delta values cannot be calculated at the beginning of the invocation. The profile data output at the end of command invocation is controlled by thresholds. The /floff/fP value disables profile data output. If profile data annotation is enabled but profile metric gathering has not been enabled with set_trace_option, all profile metrics are auto-enabled. If omitted, annotation is started without profile data output.

-log *log_string*

This option outputs the given log_string to the shell output prepended with ";## ". If the given string contains newline characters, each line is prendended with ";## ".

-quiet

If given, this option causes the command to ignore error conditions such as an attempt to log a comment string before annotation is started. The command will exit quietly when it encounters an error condition.

DESCRIPTION

The **annotate_trace** command performs operations related to annotating traced command execution to the shell output. If the application creates an output log, the annotation is made also to the output log. The command strings which are annotated to the output are those that are traced for creating a traced command replay log.

The **annotate_trace** with no options will report on the current status of annotation: on or off.

EXAMPLES

The following example starts annotation of executed commands to the output. Then the command is invoked again with no options to report on the status.

```
prompt> annoate_trace -start
prompt> annoate_trace
on
```

The following example configures profile metrids, then starts annotation of executed commands to the output along with the profile data.

```
prompt> set_trace_topion -profile all
prompt> set_trace_topion -memory_threshold 10
prompt> set_trace_topion -cpu_threshold 0
prompt> annotate_trace -start -profile with_start
```

SEE ALSO

set_trace_option(2)
get_trace_option(2)
log_trace(2)

append_to_collection

Adds objects to a collection and modifies a variable.

SYNTAX

collection **append_to_collection**
[-unique]
var_name
object_spec

Data Types

var_name	collection
object_spec	list

ARGUMENTS

-unique

Indicates that duplicate objects are to be removed from the resulting collection. By default, duplicate objects are not removed.

var_name

Specifies a variable name. The objects matching *object_spec* are added into the collection referenced by this variable.

object_spec

Specifies a list of named objects or collections to add.

DESCRIPTION

The **append_to_collection** command allows you to add elements to a collection. This command treats the variable name given by the *var_name* option as a collection, and it appends all of the elements in *object_spec* to that collection. If the variable does not exist, it is created as a collection with elements from the *object_spec* as its value. If the variable does exist and it does not contain a collection, it is an error.

The result of the command is the collection that was initially referenced by the *var_name* option, or the collection created if the variable did not exist.

The **append_to_collection** command provides the same semantics as the common use of the **add_to_collection** command; however, this command shows significant improvements in performance.

An example of replacing the **add_to_collection** command with the **append_to_collection** command is provided below. For example,

```
set var_name [add_to_collection $var_name $objs]
```

Using the **append_to_collection** command, the command becomes:

```
append_to_collection var_name $objs
```

The **append_to_collection** command can be much more efficient than the **add_to_collection** command if you are building up a collection in a loop. The arguments of the command have the same restrictions as the **add_to_collection** command. For more information about these restrictions, see the **add_to_collection** man page.

EXAMPLES

The following example from PrimeTime shows how a collection can be built up using the **append_to_collection** command:

```
pt_shell> set xports
Error: can't read "xports": no such variable
    Use error_info for more info. (CMD-013)
pt_shell> append_to_collection xports [get_ports in*]
{"in0", "in1", "in2"}
pt_shell> append_to_collection xports CLOCK
{"in0", "in1", "in2", "CLOCK"}
```

SEE ALSO

add_to_collection(2)
foreach_in_collection(2)
index_collection(2)
remove_from_collection(2)
sizeof_collection(2)

apply_clock_gate_latency

Annotates the clock latencies on the existing clock-gating cells based on the settings previously specified using the **set_clock_gate_latency** command.

SYNTAX

status **apply_clock_gate_latency**

ARGUMENTS

The **apply_clock_gate_latency** command has no arguments.

DESCRIPTION

The **apply_clock_gate_latency** command annotates the clock latencies on the existing clock-gating cells based on the settings previously specified using the **set_clock_gate_latency** command.

Use this command to manually annotate clock-latency values specified by the **set_clock_gate_latency** command to existing clock-gating cells.

The tool only annotates the latency value on the clock pin of a cell (clock gate or register) whose stage is N and whose clock domain is *clk*, if the **set_clock_gate_latency -stage N** was explicitly used for the clock domain. The tool never overwrites latency values on pin objects that are annotated by the **set_clock_latency command**, but it can overwrite values that were annotated due to the **set_clock_gate_latency** command or by optimizations that were enabled by the application option **compile.clockgate.physically_aware**. Latency annotation is only supported for flip-flops and clock gates. The tool does not annotate macro cell pins.

To remove annotated latencies that were specified using the **set_clock_gate_latency** command, use the **reset_clock_gate_latency** command.

To remove the clock gate latency settings that were specified by the **set_clock_gate_latency** command in the current scenario, use the **set_clock_gate_latency -reset** command.

To report the clock gate latency settings that were specified by the **set_clock_gate_latency** command, use the **report_clock_gate_latency** command.

Multicorner-Multimode Support

The actual clock latency annotation performed during the execution of **compile** or **apply_clock_gate_latency** commands is done for all the scenarios for which a clock latency value is available.

SEE ALSO

compile.clockgate.physically_aware(3)
report_clock_gate_latency(2)
reset_clock_gate_latency(2)
set_clock_gate_latency(2)
set_clock_latency(2)

apply_power_model

Apply a power model which describes the power intent of a reference library cell to instances.

SYNTAX

string **apply_power_model**
[*power_model_name*]
[-elements *instance_names*]
[-supply_map *supply_map_list*]
[-port_map *port_map_list*]
[-parameters *param_map_list*]
[-all_hard_macros]

Data Types

power_model_name string
instance_names list of instances
supply_map_list list of {power_model_supply_set current_scope_supply_set}
port_map_list list of {power_model_supply/logic_port current_scope_supply/logic_net}
param_map_list list of {power_model_parameter parameter_value}

ARGUMENTS

power_model_name

Specifies the name of the power model to be applied. The name should be a simple (non-hierarchical) name.

The applied power model is searched in this descending order of preference:

1. Power models defined in the current scope.
2. Power models defined in any direct ancestor scope: the closest the better.
3. Power models defined in user power model library. Please see app options mv.upf.power_model_search_path(3) and mv.upf.power_model_library(3).
4. Power models defined in any scope.

-elements *instance_names*

Specifies a collection of cells that this power model should be applied to.

If this option is not specified, the power model will be applied to all cells in and under the current scope whose reference library cells match those specified in the power model (i.e. library cells listed in **-for** option of the **define_power_model** command).

-supply_map *supply_map_list*

Specifies the mapping of supply sets in the power model to supply sets in the current scope. For example:

-supply_map {{model_vdd top_vdd} {model_vss top_vss}}

-port_map ***port_map_list***

Specifies the mapping of supply/logic ports in the power model to supply/logic nets in the current scope. For example:

-port_map {{model_supply_port top_supply_net} {model_logic_port top_logic_net}}

-parameters ***param_map_list***

Specifies the mapping of parameters in the power model to actual parameter values. For example:

-parameters {{model_param1 "value_1"} {model_param2 "value_2"}}

-all_hard_macros

Apply power models to all hard macro cells under the current scope with matching power model, i.e., the macro lib cell is included in the **-for** option of the **define_power_model** command, or the power model name is the lib cell name.

power_model_name must not be provided and no other options can be specified.

DESCRIPTION

The apply_power_model command binds the power_model_name to the cell instances specified by the -elements option. The associated parameter binding specified via the -parameters option allows objects within the environment to be bound to parameters defined within the power model. Not all parameters within the power model need to be bound to objects in the environment and parameters that are not bound shall retain their default values during run time.

The apply_power_model command sets the scope to each of the specified set of instances, referred to as the instance scope, and then executes the set of UPF commands in the power model power_model_name. Upon return, the current scope is restored to the scope in which the command was invoked.

EXAMPLES

The following example applies a power model to two macro cells:

```
prompt> define_power_model MODEL -for {macro_lib_cell} {
  create_supply_set SS_macro
  create_supply_set SS_macro_ao
  create_supply_port VDD_macro
  create_supply_port VSS_macro
}
prompt> create_supply_set SS_top
prompt> create_supply_set SS_top_ao
prompt> create_supply_net VDD_top
prompt> create_supply_net VSS_top
prompt> apply_power_model MODEL -elements {u1/macro u2/macro}
                        -supply_map {{SS_macro SS_top} {SS_macro_ao SS_top_ao}}
                        -port_map {{VDD_macro VDD_top} {VSS_macro VSS_top}}
```

The following example applies a power model to all instanced cells of lib_cell in or under the current scope. Power domain

PD_macro will be created in each of the macro cells:

```
prompt> define_power_model MODEL -for {lib_cell} {
  add_parameter DOMAIN -default domain -description "top power domain in model"
  create_supply_net VDD
  create_supply_net VSS
  create_supply_set SS -function {power VDD} -function {ground VSS}
  create_power_domain PD_$DOMAIN -supply {primary SS} -include_scope
}
prompt> apply_power_model MODEL -parameters {{DOMAIN macro}}
```

The following example applies power model MODEL_A to all instantiated cells of lib_cell_A and applies power model lib_cell_B to all instantiated cells of lib_cell_B in or under the current scope:

```
prompt> sh cat ./power_model.upf
define_power_model MODEL_A -for {lib_cell_A} { create_power_domain PD_A }
prompt> sh cat ./lib_cell_B.upf
define_power_model lib_cell_B { create_power_domain PD_B }
prompt> set_app_options -as_user_default \
   -name mv.upf.power_model_search_path -value "."
prompt> set_app_options -as_user_default \
   -name mv.upf.power_model_library -value "power_model.upf lib_cell_B.upf"
prompt> apply_power_model -all_hard_macros
```

SEE ALSO

define_power_model(2)
report_power_model(2)
add_parameter(2)
mv.upf.power_model_search_path(3)
mv.upf.power_model_library(3)
mv.upf.power_model_verbose(3)

apropos

Searches the command database for a pattern.

SYNTAX

string **apropos**
  [-symbols_only]
  pattern

Data Types

pattern string

ARGUMENTS

-symbols_only

Searches only command and option names.

pattern

Searches for the specified *pattern*.

DESCRIPTION

The **apropos** command searches the command and option database for all commands that contain the specified *pattern*. The *pattern* argument can include the wildcard characters asterisk (*) and question mark (?). The search is case-sensitive. For each command that matches the search criteria, the command help is printed as though **help -verbose** was used with the command.

Whereas **help** looks only at command names, **apropos** looks at command names, the command one-line description, option names, and option value-help strings. The search can be restricted to only command and option names with the **-symbols_only** option.

When searching for dash options, do not include the leading dash. Search only for the name.

EXAMPLES

In the following example, assume that the **get_cells** and **get_designs** commands have the **-exact** option. Note that without the **-symbols_only** option, the first search picks up commands which have the string "exact" in the one-line description.

```
prompt> apropos exact
 get_cells          # Create a collection of cells
  [-exact]           (Wildcards are considered as plain characters)
  patterns           (Match cell names against patterns)

 get_designs        # Create a collection of designs
  [-exact]          (Wildcards are considered as plain characters)
  patterns           (Match design names against patterns)

 real_time          # Return the exact time of day

prompt> apropos exact -symbols_only
 get_cells          # Create a collection of cells
  [-exact]           (Wildcards are considered as plain characters)
  patterns           (Match cell names against patterns)

 get_designs        # Create a collection of designs
  [-exact]          (Wildcards are considered as plain characters)
  patterns           (Match design names against patterns)
```

SEE ALSO

help(2)

as_collection

Find a collection by name

SYNTAX

string **as_collection** [-check]
collection1

string *collection1*

ARGUMENTS

-check

Return 1 if collection1 is a collection

collection1

The collection to convert

DESCRIPTION

This command looks up a string value and returns it as a collection it the collection is still live. All other values are passed through this command unmodified.

This command is also provides a simple way to check if the input is a collection.

EXAMPLES

Check if a value is a collection:

```
shell> as_collection -check [get_cells]
1
shell> as_collection -check hello
0
```

SEE ALSO

collections(2)
foreach_in_collection(2)

assign_3d_interchip_nets

Derives logical connections between bumps and drivers that have the shortest wirelength between them.

SYNTAX

status **assign_3d_interchip_nets**
[-bumps *cell_list*]
[-nets *net_list*]
[-matching_types *matching_types*]
[-set_port_terminals incremental | all]

Data Types

cell_list collection
net_list collection
matching_types list

ARGUMENTS

-bumps *cell_list*

Specifies the bumps for logical connection propagation. By default, all bumps in the design are considered. If only one bump is selected, its center-aligned bump is selected as a pair.

-nets *net_list*

Specifies the nets for logical connection propagation. By default, all top-level nets in design are available.

-matching_types *matching_types*

Specifies the matching types used to categorize different assignment groups. By default, all matching types are considered. If there is no matching type defined in the design, all bumps and drivers are considered in wirelength calculation.

-set_port_terminals incremental | all

Specifies the strategy to update the terminals(preferred_pin) for ports. Default value is **incremental**.

- **incremental**: only set terminals(preferred_pin) to the ports that have not been set and can be set with preferred_pin.
- **all**: set terminals(preferred_pin) to all ports no matter if they have terminals(preferred_pin) or not.

DESCRIPTION

This command builds logical connections between bumps and drivers with the shortest wirelength. The net for assignment should have the driver-pin pair connected. Meanwhile, the bump pin should be free for assignment. If there are one or more bump pins on the net, the net will be skipped. You can use the **disconnect_3d_bumps** command before performing net assignment or specifying the correct net for assignment.

EXAMPLES

The following example derives the logical connection between drivers on the net sip_data_a[19] with the bump pair {mini_cpu_chip_inst/BUMP_RING9 addmult_sip_inst/BUMP_RING9}.

```
prompt> assign_3d_interchip_nets -nets sip_data_a[19] \
  -bumps {mini_cpu_chip_inst/BUMP_RING9 addmult_sip_inst/BUMP_RING9}
```

SEE ALSO

create_3d_mirror_bumps(2)
disconnect_3d_bumps(2)
propagate_3d_connections(2)
propagate_3d_matching_types(2)
set_cell_location(2)

assign_feedthrough_supply

Used to assign related supplies for feedthough ports of a given net or net connected to a given pin.

SYNTAX

status **assign_feedthourgh_supplies**
[-net nets]
[-pin pins]

Data Types

nets collection
pins collection

ARGUMENTS

-net

generate the related supplies for the feedthrough pins connected to these given nets

-pin

generate the related supplies for the given feedthrough pins and feedthrough pins on the net connected to these pins.

DESCRIPTION

The **assign_feedthough_supplies** command determines the best supplies to use for as the related supplies for the feedthough ports on a given net or a net connected to the given pin. The supplies are determined with the goal of minimizing the use of dual rail cell buffer/inverters in the feedthough domain. Only one of the -net or -pin must be specified and at least one of them should be specified.

SEE ALSO

place_pins(2)

assign_tsv

Assigns the specified TSV (Through Silicon Via) in the TSV list to the specified logical net and pins in the pin list.

SYNTAX

int **assign_tsv**
tsv_list
[-net *net*]
[-pins *pin_list*]
[-prefix *prefix*]
[-force]

Data Types

net net
pin_list list
prefix string

ARGUMENTS

tsv_list

Specifies the TSV to assign. If a TSV list is specified, the first TSV in the list is the source point for routing. At least one TSV must be specified. This is a required option.

-net *net*

Specifies the net to assign the TSV to. All specified objects are logically connected to the net. If all objects are initially unconnected to any net, a new net will be created with the given prefix that specified with option **-prefix**. If an existing net is already connected to some of the objects, the net will be reused. To avoid user error, the command does not allow two existing nets to be assigned together. This option and the **-pin** option are mutually exclusive.

-pins *pin_list*

Specifies a list of pins to assign to the TSV. This option and the **-net** option are mutually exclusive.

-prefix *prefix*

Specifies a prefix for the name of a new net, if created.

-force

Force the TSV to connect to the specified net.

DESCRIPTION

This command assigns the specified TSV in the tsv_list to the specified logical net and pins in the pin_list.

EXAMPLES

The following example assigns the specified TSV.

```
prompt> assign_tsv tsv1
```

SEE ALSO

assign_3d_interchip_nets(2)
check_3d_design(2)
create_3d_mirror_bumps(2)
create_bond_pad_array(2)
create_bump_array(2)
create_tsv_array(2)
disconnect_3d_bumps(2)
propagate_3d_connections(2)
propagate_3d_matching_types(2)
set_cell_location(2)

associate_checkpoint_action

Associates an action with checkpoint names.

SYNTAX

integer **associate_checkpoint_action**
[-before *{list_of_checkpoint_patterns}*]
[-after *{list_of_checkpoint_patterns}*]
[-replace *{list_of_checkpoint_patterns}*]
[-enable *name_of_action*]
[-disable *name_of_action*]

Data Types

list_of_checkpoint_names list
name_of_action string

ARGUMENTS

-before *list_of_checkpoint_patterns*

Specifies that the action should run before all checkpoints matching any of the given checkpoint patterns. This option takes a Tcl list of checkpoint name patterns as its parameter. Those names can be simple checkpoint names or include glob-style * wildcard characters. Mutally exclusive with the -after and -replace options.

-after *list_of_checkpoint_patterns*

Specifies that the action should run after all checkpoints matching any of the given checkpoint patterns. This option takes a Tcl list of checkpoint name patterns as its parameter. Those names can be simple checkpoint names or include glob-style * wildcard characters. Mutally exclusive with the -before and -replace options.

-replace *list_of_checkpoint_patterns*

Specifies that the action should run as a replacement for the Tcl commands wrapped by the eval_checkpoint command. This happens for all checkpoints matching any of the given checkpoint patterns. This option takes a Tcl list of checkpoint name patterns as its parameter. Those names can be simple checkpoint names or include glob-style * wildcard characters. Mutally exclusive with the -before and -after options.

-enable *name_of_action*

Specifies the name of the action to enable at the matching checkpoint patterns of the -before, -after, or -replace option. Mutually exclusive with the -disable option. name_of_action must be an action previously defined with the create_checkpoint_action command.

-disable *name_of_action*

Specifies the name of the action to disable at the matching checkpoint patterns of the -before, -after, or -replace option. Mutually exclusive with the -enable option. name_of_action must be an action previously defined with the create_checkpoint_action command. If an action is both disabled and enabled at a checkpoint, the disable setting takes precedence and the action will not run.

DESCRIPTION

This command configures an action to run at one more more checkpoints triggered by the eval_checkpoint command in the flow scripts. Actions are defined in the checkpoint system with the create_checkpoint_action command.

This command has two required options. The first is one of -enable or -disable, which takes the name of the action as its parameter. This option specifies the name of the action being associated, and whether that action should be enabled or disabled for the patterns that will be given with the -before/-after/-replace option. -enable and -disable are mutually exclusive on a single call to associate_checkpoint_action. But multiple calls can be done to this command in sequence to give both enable and disable constraints. The -enable and -disable settings from multiple invocations of associate_checkpoint_action are time-sequenced. For example, enabling an action to run at * (all checkpoints), and then disabling it to run at the 'create_placement' checkpoint will result in that action running at all checkpoints except for 'create_placement'. This is a typical application of the disable option, where an action is enabled to run at multiple checkpoints using a wildcard pattern, but then disabled for one or more specific checkpoints that were matched by that pattern.

The second required option is one of -before, -after, or -replace. This option specifies the timing of the execution of the action, relative to the checkpointed command. Only one of -before, -after, or -replace can be given.

- -before indicates that the action should run before the checkpointed command. A typical before action is setting an app option before running a command.
- -after indicates that the action should run after the checkpointed command. A typical after action does some post-command cleanup, like changing constraints.
- -replace indicates that the action should run in place of the checkpointed command. In this case, the action is run but the original Tcl command/s wrapped by eval_checkpoint do not execute. A typical usage of a replace action is to skip execution of a command (by replacing it with nothing), or to modify the options of a command. The 'get_current_checkpoint -command' command can be run inside such a replacement action to get the original command including it's options, so that they can be modified by the action.

  Any of the -before, -after, -replace options take a required parameter which is a Tcl list of checkpoint name patterns at which to execute the action. For example, -before "*placement* place_opt_initial_place" specifies that an action should execute before any checkpoint with the name place_opt_initial_place, or with a name that includes the sub-string 'placement'.

  Action definitions and associations should not be part of the golden flow scripts for the design or project. They are intended to execute non-default actions that deviate from the standard flow. They are typically included in the ./checkpoint.config.tcl file in your run directory, which is automatically sourced by the checkpoint system.

EXAMPLES

Configures the gropto action to run before any *initial_drc checkpoints, to enable GR-based high-fanout synthesis in and place_opt

```
## In ./checkpoint.config.tcl
create_checkpoint_action gropto {
  set_app_options \
-name place_opt.initial_drc.global_route_based \
```

```
-value true
}

associate_checkpoint_action -enable gropto -before *initial_drc

## Action will execute before this line in the flow script place_opt.tcl
eval_checkpoint place_opt_to_initial_drc {
place_opt -from initial_place -to initial_drc
}
```

Configures the high_eff_cong action to force high-effort congestion on all create_placement* and refine_placement* checkpoints. get_current_checkpoint -command is used to access and modify the original command options, and the -replace option is used to replace the original command contents. ## In ./checkpoint.config.tcl

```
create_checkpoint_action high_eff_cong {
 eval [get_current_checkpoint -command] -congestion_effort high
}

associate_checkpoint_action -enable high_eff_cong \
  -replace "create_placement* refine_placement*"

## Action will execute as a replacement for checkpoints like these in the flow scripts
eval_checkpoint create_placement_incr {create_placement -incremental}
eval_checkpoint refine_placement {refine_placement -effort high}
```

SEE ALSO

create_checkpoint_action(2)
remove_checkpoint_actions(2)
create_checkpoint_report(2)
remove_checkpoint_reports(2)
associate_checkpoint_report(2)
eval_checkpoint(2)
get_current_checkpoint(2)
set_checkpoint_options(2)
reset_checkpoints(2)
get_checkpoint_data(2)

associate_checkpoint_report

Associates a report with checkpoint names.

SYNTAX

integer **associate_checkpoint_report**
[-before *{list_of_checkpoint_patterns}*]
[-after *{list_of_checkpoint_patterns}*]
[-enable *name_of_report*]
[-disable *name_of_report*]

Data Types

list_of_checkpoint_names list
name_of_report string

ARGUMENTS

-before *list_of_checkpoint_patterns*

Specifies that the report should run before all checkpoints matching any of the given checkpoint patterns. This option takes a Tcl list of checkpoint name patterns as its parameter. Those names can be simple checkpoint names or include glob-style * wildcard characters. Mutally exclusive with the -after option.

-after *list_of_checkpoint_patterns*

Specifies that the report should run after all checkpoints matching any of the given checkpoint patterns. This option takes a Tcl list of checkpoint name patterns as its parameter. Those names can be simple checkpoint names or include glob-style * wildcard characters. Mutally exclusive with the -before option.

-enable *name_of_report*

Specifies the name of the report to enable at the matching checkpoint patterns of the -before, -after, or -replace option. Mutually exclusive with the -disable option. name_of_report must be a report previously defined with the create_checkpoint_report command.

-disable *name_of_report*

Specifies the name of the report to disable at the matching checkpoint patterns of the -before, -after, or -replace option. Mutually exclusive with the -enable option. name_of_report must be a report previously defined with the create_checkpoint_report command. If a report is both disabled and enabled at a checkpoint, the disable setting takes precedence and the report will not run.

DESCRIPTION

This command configures a report to run at one more more checkpoints triggered by the eval_checkpoint command in the flow scripts. Reports are defined in the checkpoint system with the create_checkpoint_report command.

This command has two required options. The first is one of -enable or -disable, which takes the name of the report as its parameter. This option specifies the name of the report being associated, and whether that report should be enabled or disabled for the patterns that will be given with the -before/-after option. -enable and -disable are mutually exclusive on a single call to associate_checkpoint_action. But multiple calls can be done to this command in sequence to give both enable and disable constraints. The -enable and -disable settings from multiple invocations of associate_checkpoint_action are time-sequenced. For example, enabling a report to run at * (all checkpoints), and then disabling it to run at the 'load_constraints' checkpoint will result in that action running at all checkpoints except for 'load_constraints'. This is a typical application of the disable option, where an action is enabled to run at multiple checkpoints using a wildcard pattern, but then disabled for one or more specific checkpoints that were matched by that pattern.

The second required option is one of -before or -after. This option specifies the timing of the execution of the report, relative to the checkpointed command. Only one of -before or -after can be given.

- -before indicates that the report should run before the checkpointed command. A before report is used to measure QoR right before a command executes.
- -after indicates that the action should run after the checkpointed command. An after report is used to measure QoR right after a command executes.

  Both the -before and -after options take a required parameter which is a Tcl list of checkpoint name patterns at which to execute the report. For example, -after "*final route_opt*" specifies that an report should execute after any checkpoint with a name ending in 'final', as well as all route_opt checkpoints.

  Typically, report are written to files in the ./checkpoint directory, but this is not enforced and you can write reports to any disk location.

  Report definitions and associations should not be part of the golden flow scripts for the design or project. They are intended to separate reporting from the flow, such that individual users can customize their reporting needs without touching the golden flow. They are typically included in the ./checkpoint.config.tcl file in your run directory, which is automatically sourced by the checkpoint system.

EXAMPLES

Save a block before route_opt for interactive debugging. get_current_checkpoint -name is used to give the block a unique name.

```
## In ./checkpoint.config.tcl
create_checkpoint_report save_block {
 save_block -as checkpoint_[get_current_checkpoint -name]
}

associate_checkpoint_report -enable save_block -before route_opt

## Block will be saved before this line from the route_opt.tcl flow script
eval_checkpoint route_opt { route_opt }
```

Generate useful timing reports throughout the flow. Disables timing report generation at the read_constraints checkpoints. Uses the get_current_checkpoint -name and -position commands to give unique names to the reports before or after any checkpoint.

```
## In ./checkpoint.config.tcl
create_checkpint_report timing {
 set name [get_current_checkpoint -name]
 set pos [get_current_checkpoint -position]
 redirect -file ./checkpoint/$name.$pos.qor.rpt { report_qor }
 redirect -append ./checkpoint/$name.$pos.qor.rpt { report_qor -summary }
 redirect -file ./checkpoint/$name.$pos.path.rpt { report_timing -max_paths 10 }
}

associate_checkpoint_action -enable timing -before route_opt*
associate_checkpoint_action -enable timing -after *
associate_checkpoint_action -disable timing -after read_constraints
```

SEE ALSO

create_checkpoint_action(2)
remove_checkpoint_actions(2)
create_checkpoint_report(2)
remove_checkpoint_reports(2)
associate_checkpoint_action(2)
eval_checkpoint(2)
get_current_checkpoint(2)
set_checkpoint_options(2)
reset_checkpoints(2)
get_checkpoint_data(2)

associate_mv_cells

Associates the UPF strategies to the power management cells in the current design, and reports the correlation between power management cells in the netlist and the UPF strategies.

SYNTAX

status **associate_mv_cells**
[-isolation_cells]
[-level_shifters]
[-retention_registers]
[-power_switches]
[-all]
[-all_blocks | -blocks *block_designs*]
[-host_options *host_option_name*]

ARGUMENTS

-all_blocks

Associate MV cells for all the child blocks in the current design and the current design itself. This option is mutually exclusive with *-blocks*. Only one of them can be specified at a time.

-blocks *block_designs*

Specifies the list of child block designs to associate MV cells. This option is mutually exclusive with *-all_blocks*. Only one of them can be specified at a time.

-host_options *host_option_name*

Specifies the host option to use for distributed processing. If not specified, the tool uses the global host option if the option contains a distributed processing settings.

-isolation_cells

Reports the correlation and discrepancies between isolation cells and isolation strategies.

-level_shifters

Reports the correlation and discrepancies between level-shifter cells and level-shifting strategies.

-retention_registers

Reports the correlation and discrepancies between retention cells and retention strategies.

-power_switches

Reports the correlation and discrepancies between power switches and UPF power-switch strategies.

-all

Reports the correlation and discrepancies between all type of power management cells and the strategies.

DESCRIPTION

The **associate_mv_cells** command analyzes the current design and associates existing power-management cells according to the UPF strategies. It also checks and reports the correlation between the power-management cells in the netlist and the corresponding UPF strategies. The checking is based only on the UPF strategies. This command does not check for electrical violations.

If none of the power-management cell types are specified, this command associates all types of the power-management cells in the design.

EXAMPLES

The following example shows the report from the **associate_mv_cells** command when you do not specify any argument. All types of power-management cells are reported.

```
prompt> associate_mv_cells
prompt>Information: Total 223 power switch cells have been associated. (MV-021)
prompt>Information: Total 100 retention cells have been associated. (MV-021)
prompt>Information: Total 40 isolation cells have been associated. (MV-021)
prompt>Information: Total 50 level shifter cells have been associated. (MV-021)
1
```

SEE ALSO

check_mv_design(2)

associate_performance_via_ladder

Associate via rules on lib_pins.

SYNTAX

status **associate_performance_via_ladder**
-lib_pins lib_pin_names
-output association_file_name
-reset
-dont_use

Data Types

lib_pin_names string or collection of lib_pins
association_file_name string

ARGUMENTS

-lip_pins *lib_pin_names*

Specifies lib_pin_names which need performance via ladder candidates.

-output *association_file_name*

Specifies output via ladder association file name. [Default: auto_perf_via_ladder_association.tcl]

-reset

Reset existing via ladder candidates from lib_pins

-dont_use

Allow via_ladder candidates on lib_cells with dont_use attr

DESCRIPTION

This command generates association file for via rules on each lib_pin through set_via_ladder_candidate. This command returns 1 if succeeded, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the options.

EXAMPLES

The following is the usage examples.

```
prompt> associate_performance_via_ladder
1

prompt> associate_performance_via_ladder -lib_pins [get_lib_pins */*D12*/Z]
1
```

SEE ALSO

set_via_ladder_candidate(2)
reset_via_ladder_candidates(2)
report_via_ladder_candidates(2)

associate_supply_set

Associates a supply set or supply set ref to a supply set handle. The supply set handle should be a pre-defined supply set for the power domain. User defined supply sets cannot be used as supply set handle.

SYNTAX

string **associate_supply_set**
[-handle {supply_set_handle}*]
supply_set_ref

Data Types

supply_set_ref string
supply_set_handle string

ARGUMENTS

-handle {*supply_net_handle*}*

Specifies the name of the predefined supply set of the given domain. This should be the name constructed from the power domain name or strategy name. Predefined *supply_set_handles* for the power domain include: *domain_name.primary*, *domain_name.default_isolation* and *domain_name.default_retention*. This argument is required.

supply_set_ref

Specifies the supply set to be associated with the specified handle.

DESCRIPTION

The **associate_supply_set** command associates a supply set with a power domain or strategy handles. Handles are names rooted in the active scope. Handles should be predefined supply sets for the power domain. If shall be an error if :

1) A supply set handles are already associated with a supply set.

2) The associations of handles to supply sets form a loop of associations.

EXAMPLES

The following example associates a supply set named SS1 to PD.primary

```
prompt> associate_supply_set SS1 -handle PD.primary
```

SEE ALSO

create_supply_net(2)
create_power_domain(2)

attach_drc_error_data

Attaches a DRC error data object to the current block.

SYNTAX

status **attach_drc_error_data**
-name *name*
drc_error_data
[-keep]

Data Types

name string
drc_error_data collection

ARGUMENTS

-name *name*

Specifies the name for the new DRC error data file to attach to the current block. The name must be unique; the command fails if the specified name is already used by an error data file attached to the block.

drc_error_data

Specifies the collection that contains DRC error data object to attach. The collection must be derived from an external error data file, such as a previously exported error data file, a manually created error data file, or an error data file from a third-party tool.

-keep

Keeps the specified error data collection open and in memory.

By default, the command closes the specified error data collection.

DESCRIPTION

The **attach_drc_error_data** command attaches the specified error data file to the current block.

The command returns 1 if successful; 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example attaches the current physical DRC error data to a file that is named "dppinassgn.err".

```
prompt> attach_drc_error_data -name dppinassgn.err [get_drc_error_data]
1
```

SEE ALSO

create_drc_error_data(2)
open_drc_error_data(2)
close_drc_error_data(2)
save_drc_error_data(2)
get_drc_error_data(2)
remove_drc_error_data(2)
collections(2)

audit_scripts

audit_scripts

Reads in a script, examines the application options and variables used in the script, and reports any changes to those application options or variables against a specified version of the tool.

The command also reports references to .tbc files.

SYNTAX

status **audit_scripts**
-input *script_name_or_path*
[-skip_hidden]
[-from *version*]

Data Types

script_name_or_path string
version string

ARGUMENTS

-input *script_name_or_path*

Specifies the location of the user scripts you want to audit. You can specify a single Tcl file or a directory name. If you specify a directory, the command audits all files in the directory with the .tcl suffix.

The tool issues an error message if *script_name_or_path* is not a valid file or directory.

-from *version*

A string specifying the tool version to compare your scripts against.

Note that only some tool versions are available for comparison, and that these versions can change across releases. To get a current list of the tool versions available for comparison, run the following command:

```
prompt> audit_script -input . -from x
Error: Value 'x' for option '-from' is not valid. Specify one of:
   18.06-sp5, 18.06-sp4, 18.06-sp2, 17.09-sp5 (CMD-031)
```

By default, if the -from option is not specified, the command compares your script against the most current tool version available.

-skip_hidden

By default, the **audit_scripts** command reports hidden application options, variables, and Tcl commands. If you specify the -skip_hidden option, these items will not be reported.

DESCRIPTION

The **audit_scripts** command reads in one or more scripts, checks the scripts for any deprecated, removed, or otherwise changed application options and variables, and reports the results in an annotated version of each script. The annotated scripts are written to a subdirectory named ./audit, each with the same name as the original file.

An annotated script contains AUDIT_INFO and AUDIT_ERROR comments that provide information about each application option, application variable, and Tcl command change. These comments are also written to the command line output as information messages.

An annotated script might look like this:

```
# AUDIT_INFO at line 105: application option 'cts.verbose.buf' is unhidden.

# AUDIT_ERROR at line 147: application variable 'abor_enhance_inv_pair' is removed.

# AUDIT_ERROR at line 156: system-default of application option
'cts.use_global_route_atree_topology' is changed from true to atree.
```

AUDIT_INFO comments identify the following:

- Application options/variables that have become hidden or unhidden
- Application options that have been deprecated
- Application options whose scope has changed (global v. block-scoped)
- Application variables that can be replaced by application options
- References to .tbc files

AUDIT_ERROR comments identify the following:

- Application options/variables that have been removed
- Application options/variables whose data type has changed
- Application options/variables whose value, user-default value, or system-default value has changed (via an internal or user script)

Note that the **audit_scripts** command evaluates application options and variables that have been specified with the following commands only:

set_app_options
reset_app_options
get_app_option_value

get_app_options
report_app_options
help_app_options

set
set_app_var
get_app_var
report_app_var
write_app_var
printvar

The **audit_scripts** command also evaluates commands that access application variables as Tcl variables:

set *variable_name value*
echo $*variable_name*

To repair any changed application options or variables in your scripts, you may either

Manually repair the application options/variables in your original scripts

Manually repair the application options/variables in the annotated scripts, and then copy the annotated scripts to the location of your original scripts

EXAMPLES

The following example audits the input script test.tcl and writes out an annotated test.tcl script to the ./audit directory:

prompt> **audit_scripts -input test.tcl**

This second example audits all the Tcl scripts in the scripts directory and writes out the annotated scripts to the ./audit directory:

prompt> **audit_scripts -input scripts**

SEE ALSO

set_app_options(2)
get_app_option_value(2)
set_app_var(2)
get_app_var(2)

balance_clock_groups

Execute clock balance action among the clock balance groups.

SYNTAX

string **balance_clock_groups**

DESCRIPTION

This command will balance clocks insertion delay for all clock balance groups created by **create_clock_balance_group**.

SEE ALSO

create_clock_balance_group(2)
report_clock_balance_groups(2)
remove_clock_balance_groups(2)
get_clock_balance_groups(2)

change_abstract

Changes the view of the cells.

SYNTAX

status **change_abstract**
[-view design | abstract]
[-label *label_name*]
[-lib *library*]
[-reload]
[-force]
[-update_ref_libs]
-references *reference_list*
-cells *cell_list*

Data Types

label_name string
library string
reference_list list
cell_list collection

ARGUMENTS

-view design | **abstract**

Change the listed references or cells to the given view type. If the **-view** option is not given, then the new referenced block will have the same view type as the original block.

Valid arguments for **-view** are **design** and **abstract**.

-label *label_name*

Change the listed references to the block with the given label. To change to the default label, specify an empty *label_name* as **change_abstract -label ""**.

If the **-label** option is not given, the new referenced block will have the same label as the original block.

-lib *library*

Change the listed references to the block in the given library. The given library must be already opened. If the **-lib** option is not given, the new referenced block will be opened from the same library as the original block.

To change to a new block from a given library, that library must first be listed in the "ref-lib list" of the designs that contain it unless the **-update_ref_libs** option is specified. Use the **set_ref_libs** command to add the library to the ref-lib list.

-reload

Reload the blocks of the listed references from disk. The in-memory copy of the block will be discarded. This option cannot be used in combination with **-view**, **-label**, or **-lib**. The reference will point to the original block after the command is executed.

The **-reload** option is useful if you want to load a block that has recently been updated by some other processing. For example, the block may have been optimized on a distributed server.

Specifying the **-reload** option automatically turns on the **-force** option.

-force

Change the block reference, even if the ports of the new referenced block do not match the original block. By default, the **change_abstract** command issues an error message and exits.

When the new block has different ports, there is a chance that your design will be compromised. For example, if the new block is missing ports, it may be necessary to drop connections and/or constraints.

-update_ref_libs

Add the block library to the ref-lib list of the designs that contain it if it is not already in the ref-lib list. If an existing library in the ref-lib list has the same name as the block library, it is replaced with the block library.

-references *reference_list*

Specifies a list of names of referenced blocks. All cells that point to the referenced block will be changed as specified by the **-view**, **-label**, **-lib**, and **-reload** options. The list must not contain patterns or collections, only simple block names.

-cells *cell_list*

Specifies one or more physical hierarchy cells to change. You must specify the **-view** option together with the **-cells** option. The **-cells** option must not be used with **-label** or **-lib**.

DESCRIPTION

This command changes the current bound block of the given cells. The operation of the **change_abstract** command is similar to the **change_view** and **set_reference** commands, except that hierarchical design constraints that cross block boundaries are preserved. The **change_view** and **set_reference** commands discard all constraints that are inside or cross the boundary of the changed references.

The **change_abstract** command works only with "design" and "abstract" views. These two views are guaranteed to have equivalent function and equivalent port interfaces. This makes it possible to preserve constraints. If you are trying to change to or from other views, or if your port interface is different, you should use the **set_reference** command instead.

Alternative to change_abstract

The **change_view** and **set_reference** commands discard all constraints that are inside or cross the boundary of the changed references. If you use commands other than **change_abstract** you will need to reapply constraints to your design when you are done. You can do this by rereading constraints from your original SDC file.

EXAMPLES

The following example replaces the references to blocks blockA and blockB to the place_opt label of the blocks.

```
prompt> change_abstract -label place_opt -references {blockA blockB}
```

The following example replaces the design view of the given subblocks with an abstract view. Constraint that were originally applied to the subblock from the top level are preserved. The constraints that were saved with the abstract subblock are not promoted to the top level.

```
prompt> change_abstract -view abstract -cells {i_mmu i_fpu1}
```

The following example replaces all cell instances of "FPU" with an abstract view.

```
prompt> change_abstract -view abstract -references FPU
```

The following example uses the **-update_ref_libs** option to replace the reference to block "blockC" located in reference library "oldPath/refLibC" to the one located in the library "newPath/refLibC" and update the reference library list to contain "newPath/refLibC" instead of "oldPath/refLibC".

```
prompt> open_lib newPath/refLibC
prompt> change_abstract -lib newPath/refLibC -references blockC
      -update_ref_libs
```

SEE ALSO

change_view(2)
create_abstract(2)
remove_scenarios(2)
remove_modes(2)
remove_corners(2)

change_link

Changes the design to which a cell is linked.

SYNTAX

status **change_link**
[-design *design*]
cells
design_name

Data Types

cells list
design_name string

ARGUMENTS

-design *design*

Specifies the design from which to change the link for the given cell(s). If no design is specified, the current design is used.

cells

Specifies the cells in the current design for which the link is to be changed.

design_name

Specifies the name of the design to which the cells in *cells* are to be linked.

DESCRIPTION

The **change_link** command specifies a design for which the link for a cell is to be changed. Hence a cell is changed to be one occurrence of the specified design.

The cell can only be changed to a compatible design. For example, the design must have the same number of ports with the same name and direction as the cell.

Although the *design_name* argument accepts names in the format *library/library_cell*, it does not imply that the actual library cell used for the new cell will be from the specified library. The actual library cell used is determined by the current link library settings.

During **change_link** activity, all Synopsys database format link information is copied from the old design to the new design. If the old design is a synthetic module, all attributes of the old synthetic module are moved to the new link. After running the **change_link**

command, link the design with the **link_block** command.

EXAMPLES

The following example shows the use of **change_link** to change the reference of the U1 cell to BUFX3 from INVX3:

```
prompt> get_attribute [get_cells U1] ref_name
INVX3
1

prompt> change_link [get_cells U1] BUFX3
1

prompt> get_attribute [get_cells U1] ref_name
BUFX3
1
```

SEE ALSO

link_block(2)
set_reference(2)

change_names

Changes the names of ports, cells, and nets in a design by applying a set of rules previously defined by the **define_name_rules** command, or renames a specified instance.

SYNTAX

status **change_names**
[-rules *rule_name*]
[-hierarchy]
[-names_file *names_file*]
[-restore]
[-force]
[-include_sub_blocks]
[-log_changes *log_file*]
[-dont_touch *object_list*]
[-instance *instance*]
[-new_name *new_name*]
[-verbose]
[-skip_physical_only_cells]

Data Types

rule_name string
names_file string
log_file string
object_list collection
instance collection
new_name string

ARGUMENTS

-rules *rule_name*

Specifies the name of a rule set previously defined by the **define_name_rules** command, or the predefined **verilog** rule set. The command changes the names of objects in the design according to the specified rule set.

-hierarchy

Changes objects in the full hierarchy of the design and will not cross over into other physical hierarchies or sub-blocks. By default, only the objects of top logical hierarchy are changed.

-names_file *names_file*

Specifies a file of manual name changes. These name changes are not subject to any name rules, but a warning is reported if any name changes are not unique. By default, the command automatically makes name changes based on the names

mentioned in the names file. If you specify this option, the tool ignores the **-rules** option.

-restore

Reverses the changes recorded in the *names_file* during the current session and restores the contents of the file to the state it was in when opened.

-force

Force option along with *names_file* option, changes the names of regular bus members, even the names file entries makes it irregular. Once it becomes irregular, it is bitblasted. Without force option if the result of change_names with names_file makes a bus from regular to irregular, those changes are not applied.

-include_sub_blocks

By default, objects of top logical hierarchy are considered. When *-hierarchy* option is used then all objects within current physical hierarchy is considered and it doesn't cross over into other physical hierarchies or sub-blocks. With this option specified, it will cross over into other physical hierarchies also and in doing so it will consider only the bound views. If an instance in top level is bound to non-design view of a sub-block like for example, abstract, then the command will error out. This is because design view can't be derived from a non-design view and changing non-design view will cause it to go out of sync with its design view. In order to consistently rename all the design data and associated timing constraints, only design views should be used for linking. This option can't be used independently and has to be used along with *-hierarchy*. If any of the sub-blocks is not yet linked then using this option will first link those sub-blocks and then change the names. Include_sub_blocks option is not supported along with *-names_file* option.

-log_changes *log_file*

Specifies a log file in which to record the name changes. By default, no log file is written.

-dont_touch *object_list*

Specifies a list of modules that are not to be changed. You can specify any modules in the hierarchy of the current design to exclude from name changing.

-instance *instance*

Changes the name of an instance in the design. This option can be used only with the **-new_name** option, and not with other options. The command changes the instance name to the specified new name.

-new_name *new_name*

Specifies the new instance name when the **-instance** option is used. The new name may contain only alphanumeric characters and the underscore character (_) and must start with a letter. New name specified must not be the full hierarchical name, instead it should only be the leaf level name. The new name cannot be a reserved word used by Verilog, SystemVerilog, or VHDL, and must not conflict with an existing instance, port, and net name within the same design. Any leading backslash character is removed.

-verbose

Reports every name change. By default, the tool reports only a summary showing the number of name changes for each object type.

-skip_physical_only_cells

Specifies that physical only type of cells should not be considered for name changes. This is applicable for nets and ports of these type of cells as well.

DESCRIPTION

The **change_names** command changes the names of ports, cells, and nets in the current design as needed to conform to specified naming rules. By applying an appropriate set of naming rules, you can write out design data files that are compatible with the requirements of the target external tool when you use the **write_verilog**, **write_def**, or **write_gds** command.

To define a set of naming rules to be used with the **change_names** command, use the **define_name_rules** command. Alternatively, you can use the predefined **verilog** rule set.

To report the available sets of rules, including the predefined **verilog** rule set, use the **report_name_rules** command.

To preview the changes to the design that would be made by applying the **change_names** command, but without actually preforming those changes, use the **report_names** command.

To change the name of a specific instance in the design, use the **change_names** command with the **-instance** and **-new_name** options. These options cannot be used together with the other command options.

The **change_names** command can change the names of objects in design libraries only, not the names of leaf-level cells in reference libraries. To modify reference libraries, use the Library Manager tool.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

Names File Format

To specify a name change, you must specify the following four fields:

design_name object_type object_name new_name

The **design_name** field is a design currently read into the tool.

The **object_type** field is **port**, **cell**, or **net**.

The **object_name** field is the name of an object within the specified design.

The **new_name** field is the name that replaces the existing name.

If the file contains a report bar formed by min 10 dashed line (--------), the tool ignores all information above the report bar.

The tool parses all lines after the report bar (or all lines, if the file does not contain a report bar).

By default, the tool parses each line into four fields, which are separated by whitespace characters, such as blanks, tabs, or new lines.

This format matches the format of the **report_names** command output, which allows you to redirect the output of the **report_names** command to a file, edit it, and then use it as input to the **change_names** command.

The following example shows a names file format.

Design Type Object New Name

TOP cell U$1 U_1

TOP net NET_NAME_IS_WAY_TOO_LONG NET_NAME_IS_WAY_TOO

TOP net 12345 N12345

prompt> **change_names -names_file name.change -verbose**

EXAMPLES

The following commands change the names of ports, nets, and cells in the top logical hierarchy of the current design by converting all lowercase "a" characters to uppercase "A".

```
prompt> define_name_rules my_map_rule -map {{"a","A"}}
1
prompt> change_names -rules my_map_rule
 Information: Using name rules 'my_map_rule'.
 Information: 908 objects (124 ports, 144 cells, ... ) changed in design 'core1'.
1
```

The following command changes the name of instance u0_0 to u0_AB. (This command does not use any naming rules.)

```
prompt> change_names -instance u0_0 -new_name u0_AB
1
```

The following command changes the name of instance old_inst_name in hierarchy A/D to new_inst_name. (This command does not use any naming rules.)

```
prompt> change_names -instance A/D/old_inst_name -new_name new_inst_name
1
```

The following command changes the names in the design to meet Verilog naming conventions using the predefined **verilog** rule set. Using the **define_name_rules** command is not needed.

```
prompt> change_names -rules verilog
 Information: Using name rules 'verilog'.
 information: 601 objects (123 ports, ... ) changed in design 'topAC'.
1
```

Note: The predefined **verilog** rule set used in this example is somewhat different from using the **-special verilog** option in the **define_name_rules** command. The **-special verilog** option in the **define_name_rules** command implicitly invokes the following rules:

-collapse_name_space
-equal_ports_nets
-inout_ports_equal_nets
-remove_irregular_port_bus
-remove_irregular_net_bus

Using **change_names -rules verilog** invokes these same rules, but also invokes the following additional rules:

-check_internal_net_name
-check_bus_indexing
-flatten_multi_dimension_busses
Do not use reserved words (long list)
Use only alphanumeric characters
Start names with a letter

Using the **-special verilog** option in the **define_name_rules** command is more flexible because you can add more rules to your custom rule set. The **change_names -rules verilog** command cannot be customized.

For details of the exact rules invoked by the **verilog** keyword, use the **report_name_rules** command.

SEE ALSO

define_name_rules(2)
report_names(2)
report_name_rules(2)

change_reference

Rebinds hierarchical cells to a new reference cell

This command is deprecated, please use set_reference command.

SYNTAX

collection **change_reference**
[-design *reference_design*]
[-module *module_name*]
[-verbose]
cell_list

Data Types

reference_design design name or collection
module_name string
cell_list list

ARGUMENTS

cell_list

Specifies the list of hierarchical cells to be rebound to new reference. Each cell must be in scope (at or below the current instance).

-design *reference_design*

Specifies the new design to which the specified cells are to be bound. The specified cells must be currently bound to another design. The argument can be a design name or a collection of one design. The cells must be bound to a design with the same port interface as the new design. This argument is mutually exclusive with *-module*.

-module *module_name*

Specifies the name of an existing module in the current design to which the specified cells are to be bound. The specified cells must be currently bound to another module. The new module must have the same port interface as the module the cells are currently bound to. This argument is mutually exclusive with *-design*

-verbose

Prints additional information.

DESCRIPTION

The **change_reference** command changes the reference of the given list of hierarchical cells. All the specified cells must have the same current reference. The new reference and the current reference of the cells must have the same exact port interface. This means that the number of ports, port directions and port names must be the same. Further, the current and new references must be of the same *type*. This means that if the current reference of the cell is a design, the new one must also be a design and if the current reference is a logical module in the current design, the new one also must be a logical module in the current design. The new reference must already exist. Further, when the reference is a design, the reference design with the view bound currently, must exist. The view bound to the cell is not changed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following examples, change_reference is used for changing the reference of cells bound to designs and logical modules

```
prompt> change_reference -design [get_designs alu_1] u1
1

prompt> change_reference -module logical_alu {u2 u3}
2
```

SEE ALSO

commit_block(2)
get_designs(2)
set_reference(2)
uncommit_block(2)
uniquify(2)

change_selection

Changes the selection in the GUI, taking a collection of objects and changing the selection according to the type of change specified.

SYNTAX

int **change_selection**
[-name *slct_bus*]
[-replace]
[-add]
[-remove]
[-toggle]
[-type *object_type*]
collection

Data Types

slct_bus	string
object_type	list
collection	list

ARGUMENTS

-name *slct_bus*

Specifies to change the selection bus by using the value of *slct_bus*. By default, the command changes the selection bus by using the name *global*.

-replace

Replaces the current selection with the objects in the collection. This is the default behavior of the command if you do not specify any optional parameter.

-add

Adds the objects in the collection to the current selection. By default, this option is off.

-remove

Removes the objects that are specified in the collection from the current selection. By default, this option is off.

-toggle

Adds each item that is specified in the collection to the selection bus if it is not currently contained there. If it is currently contained in the selection bus, remove it. By default, this option is off.

-type *object_type*

Specifies the type to change. Only those items from the collection that are of the type specified by *object_type* are used to change the selection. The valid values are **design**, **port**, **net**, **cell**, **pin**, and **path** (timing path). By default, the command uses the entire collection.

collection

Specifies the collection of objects to use to change the selection. The type of change that is applied to the current selection with the *collection* is specified by the optional parameters listed above. By default, this option is off.

DESCRIPTION

The **change_selection** command changes the selection in the GUI. When selections are changed, the GUI updates all relevant windows to reflect it.

A collection of objects and the type of change are given as input to the command. The collection of objects might be returned as the result of another command such as the **get_designs** command. If the collection is empty and you use the **-replace** option (or let the command default by specifying no option), the current selection is cleared.

If you use the **-type** option, only the type of objects specified are used to change the current selection. For example, if you use **-type design**, the command uses only the design objects in the collection to change the current selection. If you do not use **-type** option, all objects in the collection are used to change the current selection. For example, if you use the **-add** option without using the **-type** option, all objects, regardless of their type, are added to the current selection.

For information about collections, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example replaces the current selection with the collection of design objects, regardless of type.

```
prompt> change_selection [get_designs]
```

The following example adds a cell named U5 to the selection made in the example above.

```
prompt> change_selection -add [get_cells U5]
```

The following example removes all pin objects from the current selection.

```
prompt> change_selection -remove -type pin [get_selection]
```

The following example clears the selection.

```
prompt> change_selection ""
```

The following example creates a new selection bus and adds a net named n1 to the selection.

```
prompt> set slct [create_selection_bus]
prompt> change_selection  -name $slct [get_nets 1]
```

SEE ALSO

collections(2)
create_selection_bus(2)
filter(2)
filter_collection(2)
get_selection(2)
query_objects(2)

change_selection_no_core

Indicates that objects had no core representation

SYNTAX

change_selection_no_core -name *Slct*
[-add]
[-remove]
-type *Type*
-names *NameList*

Slct *String*
Type *String*
NameList *list of strings*

ARGUMENTS

-name *Slct*

Slct specifies the name of a selection container for which changes were logged.

-add

This option is specified if the objects were added to the selection container. This is mutually exclusive with option -remove.

-remove

This options is specified if the objects were removed from the selection container. This is mutually exclusive with option -add.

-type *Type*

Type is the type of the objects the names of which are specified with *NameList*.

-names *NameList*

NameList is the list of names of the involved objects. All the objects are of type *Type*.

DESCRIPTION

When the change of contents of a selection container gets logged it can happen that not all the objects involved in the change can be represented by the core. If this happens then this command is logged for the objects that did not have a core representation. For each type with objects without core representation a separate command is logged. Also a separate command is logged for objects

that were added and objects that were removed. The arguments of the logged command specify which selection container changed, if the objects were added or removed, the type of the objects and their string representations.

Having this command logged documents that the contents of the log file is incomplete and a replay might not work correctly.

Depending on variable *selection_logging_no_core_action* the command behaves differently.

If the value of the variable is the string "clear" the command clears the selection container *Slct*.

If the value of the variable is the string "error" the command raises a Tcl error.

In all other cases the command does nothing.

SEE ALSO

change_selection_too_many_objects(2)

change_selection_too_many_objects

Indicates that too many objects were involved in selection change

SYNTAX

change_selection_too_many_objects -name *Slct*
[-add]
[-remove]

Slct *String*

ARGUMENTS

-name *Slct*

Slct specifies the name of a selection container for which changes were logged.

-add

This option is specified if the objects were added to the selection container. This is mutually exclusive with option -remove.

-remove

This options is specified if the objects were removed from the selection container. This is mutually exclusive with option -add.

DESCRIPTION

When the change of contents of a selection container gets logged it can happen that the number of objects involved in the change exceeds the number specified by the preference of category *Globals* and key *selection_loggin_max_names*. In this case not all objects involved in the change are logged but instead this command is logged. A separate command is logged if the number of added objects or the number of removed objects exceed the given limit. The arguments of the logged command specify which selection container changed and if the objects were added or removed.

Having this command logged documents that the contents of the log file is incomplete and a replay might not work correctly.

Depending on variable *selection_logging_too_many_objects_action* the command behaves differently.

If the value of the variable is the string "clear" the command clears the selection container *Slct*.

If the value of the variable is the string "error" the command raises a Tcl error.

In all other cases the command does nothing.

SEE ALSO

change_selection_no_core(2)

change_view

Changes the view of the cells.

Changes the view of the cells.

SYNTAX

status **change_view**
[-view *view_type*]
[-label *label_name*]
[-lib *library*]
[-reload]
[-force]
[-update_ref_libs]
-references *reference_list*
-cells *cell_list*

Data Types

view_type string
label_name string
library string
reference_list list
cell_list collection

ARGUMENTS

-view *view_type*

Specifies the name of view type to use.

If you do not specify the **-view** option, the command resets the reference view of the specified cells to the original view set by the linker and the view switch list.

Valid values for the *view_type* argument are **abstract**, **conn**, **design**, **error**, **frame**, **layout**, **outline**, and **timing**.

-label *label_name*

Change the listed references to the block with the given label. To change to the default label, specify an empty *label_name* as **change_abstract -label ""**.

If the **-label** option is not given, the new referenced block will have the same label as the original block.

-lib *library*

Change the listed references to the block in the given library. The given library must be already opened. If the **-lib** option is not

given, the new referenced block will be opened from the same library as the original block.

To change to a new block from a given library, that library must first be listed in the "ref_lib list" of the designs that contains it. unless the **-update_ref_libs** option is specified. Use the **set_ref_libs** command to add the library to the ref_lib list.

-reload

Reload the blocks of the listed references from disk and discard the in-memory copy of the block. This option cannot be used in combination with **-view**, **-label**, or **-lib**. The reference will point to the original block after the command is executed.

The **-reload** option is useful if you want to load a block that has recently been updated by some other processing. For example, the block may have been optimized on a distributed server.

Specifying the **-reload** option automatically enables the **-force** option.

-force

Change the block reference, even if the ports of the new referenced block do not match the original block. By default, the **change_abstract** command issues an error message and exits.

When the new block has different ports, there is a chance that your design will be compromised. For example, if the new block is missing ports, it may be necessary to drop connections and/or constraints.

-update_ref_libs

Add the block library to the ref-lib list of the designs that contain it if it is not already in the ref-lib list. If an existing library in the ref-lib list has the same name as the block library, it is replaced with the block library.

-references ***reference_list***

Specifies a list of names of referenced blocks. All cells that point to the referenced block will be changed as specified by the **-view**, **-label**, **-lib**, and **-reload** options. The list must not contain patterns or collections, only simple block names.

-cells ***cell_list***

Specifies one or more physical hierarchy cells to change. You must specify the **-view** option together with the **-cells** option. The **-cells** option must not be used with **-label** or **-lib**.

DESCRIPTION

This command changes the current bound view for one or more cell instances.

If the specified view type does not exist for one or more of the cell instances, the command returns an error status and does not change the bound view for those cells. However, the command does change the bound view for the other cells.

The bound view for cells in the top block is persistent. The bound view for cells in child blocks is not persistent. After a block hierarchy is saved and restored, the bound views in the top block are preserved and the bound views for cells within child blocks revert to the binding saved for that cell within that child block.

CAUTION

The **change_view** command is not intended for use on designs that have top-down constraints applied. If you use the **change_view** command, constraints that apply to the block or span the block boundary are lost and you must reapply your constraints from an SDC file. If your design has top-down constraints, you should use the **change_abstract** command.

Once all the invocations of **change_view** are done, a follow up call to the command **link_block -force** is to be done in some cases. This should be done when the block to which the cell is to be changed is either of **module** or **blackbox** design type.

EXAMPLES

The following example replaces the existing design view with the outline view for the cell instance named u1/mem1:

```
prompt> change_view -view outline u1/mem1
```

The following example specifies that the design view is to be used for the cell instance named u1/mem1:

```
prompt> change_view -view design u1/mem1
```

The following example specifies that the view switch list should be used to determine the bound view for the cell instance named u1/mem1:

```
prompt> change_view u1/mem1
```

The following example specifies that the abstract view is to be used for all cells that reference the block named MIDDLE:

```
prompt> change_view -view abstract -references MIDDLE
```

The following example uses the **-update_ref_libs** option to replace the reference to block "blockA" located in reference library "oldPath/refLibA" to the one located in the library "newPath/refLibA" and update the reference library list to contain "newPath/refLibA" instead of "oldPath/refLibA".

```
prompt> open_lib newPath/refLibA
prompt> change_view -lib newPath/refLibA -references blockA
      -update_ref_libs
```

SEE ALSO

change_abstract(2)
create_frame(2)
get_cells(2)
get_view_switch_list(2)
link_block(2)
read_verilog_outline(2)
set_reference(2)
set_view_switch_list(2)

characterize_block_pg

Writes out power and ground creation constraints, including via master rule, pattern, strategy, and strategy via rules, for each block in preparation for distributed PG creation. The constraints are based on the top-level power ground creation constraints in the hierarchical design. The command creates a power ground constraint script for each block with the same reference design. The script can be used to regenerate power and ground at the block level. The scripts are created under the pgroute_output directory in the current path. The command writes out a mapping file when you use the **-compile_pg_script** option. The map file specifies the PG constraints and script to use with the **compile_pg** command during distributed PG creation.

SYNTAX

status **characterize_block_pg**
[-output_directory *directory_name*]
[-compile_pg_script *file_name*]

Data Types

directory_name string
file_name string

ARGUMENTS

-output_directory *directory_name*

Specifies the output directory name for the generated block-level PG script files. The directory is created under the current path. If not specified, "pgroute_output" is used as the output directory.

-compile_pg_script *file_name*

Specifies the **compile_pg** script file name. The script can contain PG route application options and **compile_pg** commands, but should not include PG via master rule, pattern, strategies and strategy via rule commands.

When this option is specified, the command writes out a PG mapping file called "pg_mapfile" into the output directory specified by the **-output_directory** option. This mapping file specifies the PG constraint and PG **compile_pg** script for each block, and is needed for distributed PG creation. The PG constraint for the current top-level design is not included inside the mapping file since the top-level PG constraint has already been defined and is persistent.

By default, the mapping file is not generated. A sample map file is as follows:

```
leon3s    PG_CONSTRAINT  ./leon3s_pg.tcl
leon3s_2  PG_CONSTRAINT  ./leon3s_2_pg.tcl
leon3s_3  PG_CONSTRAINT  ./leon3s_3_pg.tcl

leon3s    COMPILE_PG     ./compile_pg_script.tcl
leon3s_2  COMPILE_PG     ./compile_pg_script.tcl
leon3s_3  COMPILE_PG     ./compile_pg_script.tcl
```

```
leon3mp   COMPILE_PG    ./compile_pg_script.tcl
```

DESCRIPTION

This command characterizes PG creation constraints, including via master rule, pattern, strategy, and strategy via rule for each block based on the top-level constraints in the hierarchical design.

This command writes out a map file that contains a list of PG constraint files and block names for each block. Before proceeding with distributed PG creation with the **run_block_compile_pg** command, the mapping file name must be assigned with the **set_constraint_mapping_file** command.

EXAMPLES

The following example characterizes PG creation constraints for all-level of physical hierarchy in the hierarchical design.

```
prompt> characterize_block_pg -compile_pg_script pg_script.tcl
```

The following example characterizes PG creation constraints for up-to second level of physical hierarchy in the hierarchical design.

```
prompt> set_editability -value false -from_level 3
prompt> characterize_block_pg -compile_pg_script pg_script.tcl
```

SEE ALSO

compile_pg(2)
create_pg_composite_pattern(2)
create_pg_macro_conn_pattern(2)
create_pg_mesh_pattern(2)
create_pg_ring_pattern(2)
create_pg_std_cell_conn_pattern(2)
create_pg_wire_pattern(2)
run_block_compile_pg(2)
set_constraint_mapping_file(2)
set_editability(2)
set_pg_strategy(2)
set_pg_strategy_via_rule(2)
set_pg_via_master_rule(2)
create_via_def(2)

characterize_topology_plans

Pushes down topology plans, nodes, edges, and repeaters into blocks within the hierarchy as needed.

SYNTAX

string **characterize_topology_plans**
[*topology_plan_list*]

Data Types

topology_plan_list collection

ARGUMENTS

topology_plan_list

Specifies a list of topology plans to characterize. The list can contain topology plan names, patterns, or collections. A collection can be specified by using the **get_topology_plans** command.

DESCRIPTION

The **characterize_topology_plans** command pushes down topology plans, nodes, edges, and repeaters into blocks within the hierarchy. If *topology_plan_list* is specified, those topology plans are characterized. If it is not specified, then all topology plans in the current block are characterized.

EXAMPLES

The following example pushes down topology plans, nodes, edges, and repeaters into blocks for the current topology plan.

prompt> **characterize_topology_plans [current_topology_plan]**

SEE ALSO

optimize_topology_plans(2)

propagate_topology_plans(2)

check_3d_design

Checks a 3D-IC design for physical and electrical connectivity. The command checks connectivity and chip design information inside the 3D IC and between two neighboring stacked chips.

SYNTAX

status **check_3d_design**
[-chip_placement]
[-physical_contact]
[-logical_physical_consistency]
[-matching_type]
[-bump_cluster]
[-physical_design_rule]
[-verbose]
[-error_view]

ARGUMENTS

-chip_placement

Checks the vertical level and overlap of the top-level chip. If two chips are specified with the same z (vertical level) value, they should not overlap. If two chips are neighboring, the chip with the larger virtual top-level area should contain the other one.

-physical_contact

Checks the correctness of physical contacted bumps. Inter-chip physical contact information includes the following:

- Non-aligned bump pairs
- Bumps that do not physically connect to a bump on the adjacent side of a vertical neighboring chip
- Bumps that contact more than one bump of a neighboring chip
- Overlap of a bump and another bump in a single chip

-logical_physical_consistency

Checks the logical and physical consistency of the design. If the top-level port instance pair does not have a corresponding physical bump pair, the command issues an error message.

-matching_type

Checks the matching type mismatch on logical connection and physical contact among all dies in the 3D design.

-bump_cluster

Checks the bump cluster in design for the following conditions:

- Bump cluster in a chip should has no more than 1 nets;
- If one bump in the physical contact pair belongs to a cluster, the other bump in the physical contact pair should also belong to another cluster.

-physical_design_rule

Checks the following constraints:

- TSV to TSV spacing constraint, which use the "minCutSpacing" defined in the tech file
- TSV to cell boundary spacing constraint, which use the "tsvKeepoutSpacing" defined in the tech file

-verbose

Prints detailed information. If you omit this option, the command prints only part of messages and a summary of the information.

-error_view

Specify the file name that would be used for GUI error browser. The default file name is "check3dDesign.err".

DESCRIPTION

In a 3D-IC, multiple chips are stacked vertically or placed side by side on a silicon interposer. At the top-level of a 3D-IC, the entire chip stack is modeled as a virtual top-level design and the individual chips are modeled as cells. If you specify **-physical_contact**, the command checks the physical connections between every neighboring chip and ensures that the contacting bumps are in the correct alignment. The command checks the elevation of each chip based on the Z-value specified by the **set_cell_location** command. Chips with the same Z-value are considered to be at the same elevation and are checked for chip overlaps. Chips with adjacent Z-values are considered to be neighboring chips and are checked for physical contact.

Note: The **check_3d_design** command should be used at the last step of 3D-IC design flow and run at the top level.

The **check_3d_design** command writes warning and error messages that indicate potential problems in the 3D-IC design. These problems might need to be corrected before manufacturing the 3D design. The command checks the following conditions:

- Cannot get the chip placement information
- Vertical level (Z-value) of each chip
- Chip coordinates at the top level
- Z-values of chips that do not start from 0
- Z-values of chips that are not continuous
- Chips on same level (with same Z-value) that are overlapped
- Two vertical neighboring chips are partially overlapped rather than the large chip completely covering the small chip
- Bumps are overlapped in the chip
- Non-aligned bump pairs
- Bumps do not physically connect to bump on adjacent side of vertical neighboring chip
- Bumps contact to more than one bumps of neighboring chip
- There is no bump pair on the adjacent sides between two vertical neighboring and overlap chips

- The physical bump connections between two contacting chips do not match the logical netlist specification
- Logically connected pins of two chips on top-level design that do not have physical connected bump pairs
- Matching type mismatch between bump pairs
- Logical connected bumps or drivers on same net have different matching types
- Bump clusters have more than 1 nets
- Physical contacts are cluster inconsistent
- Chips with net to package does not have zValue = 0
- Nets on top level does not link 2 terminals
- Spacing between TSV and chip boundary violate tsvKeepoutSpacing constrain defined in tech file
- Spacing between TSV and TSV violate minCutSpacing constrain defined in tech file

To ensure that the 3D-IC flow is successful, you should resolve any reported errors. See the man pages for the reported error message to get more information and guidance.

Prerequisites

create_3d_mirror_bumps
propagate_3d_connections
propagate_3d_matching_types
set_cell_location

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example performs all 3D-IC checks on the design.

prompt> **check_3d_design**

The following example command performs a physical contact bumps check of the stacked chips.

prompt> **check_3d_design -physical_contact**

SEE ALSO

create_3d_mirror_bumps(2)
propagate_3d_connections(2)
propagate_3d_matching_types(2)
report_3d_chip_placement(2)
set_cell_location(2)

check_boundary_cells

Checks boundary cell placement violations in a design.

SYNTAX

status **check_boundary_cells**
[-error_view *view_name*]
[-voltage_area {*objects*}]

Data Types

view_name string
objects list

ARGUMENTS

-error_view *view_name*

Specifies the name of the generated error view. The command saves the error view data to a file in XML format with the specified name. No attachment is created nor saved in to the design database. If the view_name file already exists under the current run directory, the command overwrites the file. If a file extension is specified, it must be ".err", otherwise it is ignored. If no extension is specified, the ".err" extension is appended to the file name.

-voltage_area {*objects*}

Specifies the voltage areas to check boundary cells.

DESCRIPTION

This command checks the boundary cell placement violations according to the boundary cell rules set by *set_boundary_cell_rules*. The violations can be classified as:

1. Continuity Violation: For this rule, all the boundary cells should be a complete circuit. If no, there will be a violation. If user only insert boundary cells along all the left and right edges, the boundary cells should cover all the left and right edges. If no, there will be a violation.

2. Corner And Boundary Cell Violation: This rule requires that all the corner and boundary cells are at the correct location.

3. Orientation Violation: This rule requires that all the corner and boundary cells are at the correct orientation.

4. Short Edge And Segment Violation: This rule requires that the length of each horizontal edge of every macro (including hard keepout margin), hard placement blockage, voltage area (including guard band) must be greater than the threshold specified by the

option *-min_horizontal_jog*. If not, there will be a violation. This rule also requires that the length of each segment must be greater than the threshold specified by the option *-min_row_width*. If not, there will be a violation.

This command does not check if there is no_1x violation and does not honor the following 4 options of the **set_boundary_cell_rules** command:

- -tap_distance
- -prefix
- -separator
- -do_not_swap_top_and_bottom_inside_corner_cell

EXAMPLES

The following example checks if there is any continuity violation, correct corner and boundary cell violation or correct orientation violation in current boundary cell placement.

```
prompt> set_boundary_cell_rules -left_boundary_cell myLib/CellLeft \
        -right_boundary_cell myLib/CellRight \
        -top_boundary_cells myLib/CellTop \
        -bottom_boundary_cells myLib/CellBottom
prompt> check_boundary_cells
```

SEE ALSO

set_boundary_cell_rules(2)
remove_boundary_cell_rules(2)
report_boundary_cell_rules(2)
compile_boundary_cells(2)
compile_targeted_boundary_cells(2)
check_targeted_boundary_cells(2)

check_bufferability

Checks for a buffering supply, available lib_cells, and bufferability problems on a net segment

SYNTAX

status **check_bufferability**
[-nets *net_list*]
[-driver *pin_or_port*]
[-loads *pins_or_ports*]
[-hierarchy *logical_hierarchy_name*]
[-voltage_area *voltage_area_name*]
[-voltage_area_shape *voltage_area_shape_name*]
[-coord *location*]
[-verbose]
[-nosplit]

Data Types

net_list collection
pin_or_port collection
pins_or_ports collection
logical_hierarchy_name collection
voltage_area_name collection
voltage_area_shape_name collection
location point

ARGUMENTS

-nets *net_list*

Specify net segments to check their bufferability. It can be a net name or collection. The nets must exist in the design. If this option is used, the **-driver, -loads,** and **-hierarchy** options are not needed and are ignored if specified. This command checks bufferability for only these net segments. Other segments of the same net might have different bufferability results.

-driver *pin_or_port*

Specify a single physical net driver; the command checks the bufferability between the driver and a set of loads in a specified hierarchy. The driver should be a leaf cell output pin, an input port of the current design, or a bidirectional leaf pin or port. It can be specified as a name or collection. The driver must exist in the design. If this option is specified, **-hierarchy** must also be specified. The driver does not have to be logically connected to the specified **-loads**, nor related to the specified **-hierarchy**. This option is ignored if used together with the **-nets** option. Without **-loads** option, it will check bufferability in target hierarchy with no loads.

-loads *pins_or_ports*

Specify one or more physical net loads; the command checks the bufferability between the loads and a driver in a specified hierarchy. The loads should be leaf cell input pins, output ports of the current design, or bidirectional leaf pins or ports. The loads can be specified as a list of names or as a collection. The loads must exist in the design. If this option is specified, **-driver** and **-hierarchy** must also be specified. The loads do not have to logically connected by the same net to each other or to the specified **-driver**, nor do they have to be related to the specified **-hierarchy**. This option is ignored if used together with the **-nets** option.

-hierarchy ***logical_hierarchy_name***

Specify a logical hierarchy in which to check bufferability between a specified driver and loads. Only logical hierarchies are accepted, any leaf cell objects used with the **-hierarchy** option results in an error. The logical hierarchy can be specified as a name or a collection. The hierarchy must exist in the design. This option should be used together with the **-driver** option. The hierarchy does not have to be logically related to the specified **-driver** and **-loads**. This option is ignored if used together with the **-nets** option.

-voltage_area ***voltage_area_name***

Specify a voltage area in which to check bufferability. This option is optional. The voltage area can be specified as a name or collection. The voltage area must exist in the design. The voltage area does not have to be related to the hierarchy of any of the specified **-nets, -driver, -loads,** or **-hierarchy**. If this option is not specified, it will find the native voltage area for the given nets or the given hierarchy. The option **-voltage_area**, **-voltage_area_shape** and **-coord** are mutually exclusive to each other.

-voltage_area_shape ***voltage_area_shape_name***

Specify a voltage area shape in which to check bufferability. This option is optional. The voltage area shape can be specified as a name or collection. The voltage area shape must exist in the design. If secondary PG constraints are defined and a dual-rail cell is needed, the check will determine if the given VA shape has a strap available as per the secondary PG constraints. The option **-voltage_area**, **-voltage_area_shape** and **-coord** are mutually exclusive to each other.

-coord ***location***

This option is optional. It takes a point consisting of a tcl list of x coordinate and y coordinate. If secondary PG constraints are defined and a dual-rail cell is needed, the check will determine if the given point has a strap available as per the secondary PG constraints. The option **-voltage_area**, **-voltage_area_shape** and **-coord** are mutually exclusive to each other.

-verbose

Specify that verbose output be generated. By default, **check_bufferability** gives basic information about bufferability, including the available buffering supply, number of type of available lib_cells, and whether any bufferability problems exist. In verbose mode, detailed information is reported regarding how the buffering supply is determined. Also, it prints a full list of available and unavailable lib_cells, including reasons that those lib_cells are unavailable.

-nosplit

Specify nosplit style reporting behavior. By default, line wrapping of long rows of data is done automatically. In nosplit mode, line wrapping is not done. Readable column alignment is still maintained in nosplit mode.

DESCRIPTION

=== Summary ===

The **check_bufferability** command checks net bufferability. Use this command to understand the following information about a net:

- What supply net is used for buffering
- Whether single rail, dual rail, or insulated dual rail buffers are used for buffering
- How many and which buffer and inverter lib_cells are available for buffering the net

- Whether there are design constraints or other factors that would prevent buffering, despite an available buffering supply and available lib_cells

=== Use Models ===

There are two use models for calling **check_bufferability**:

check_bufferability -nets [-voltage_area|-voltage_area_shape|-coord]

check_bufferability -driver [-loads] -hierarchy [-voltage_area|-voltage_area_shape|-coord]

In the first use model, the **-nets** option specifies net segments on which to check bufferability. A net might consist of many logical segments, which can be accessed using **get_nets -segments**. Bufferability can be different for each net segment, so it is important to specify the correct net segment to get back correct bufferability results. To get a full understanding of the bufferability of the entire net, it might be necessary to call **check_bufferability** multiple times on different net segments. Reasons that bufferability can change for different segments of the same net include:

- The net crosses into different power domains and voltage areas. Each power domain can have different primary and available supplies. So, the net segments of one power domain might be bufferable using single rail buffers with the domain primary supply, but dual rail buffers are needed for the segments of a different power domain.
- The fanouts of the net segments change. For example, in a switched domain, one net segment might fanout to a switched load and an always-on load, requiring dual rail buffering to reach the always-on load. But, querying a different net segment that only fans out to the switched load shows that single-rail buffers can be used.
- Design constraints are present that affect net segments or hierarchies differently. For example, **set_dont_touch** constraints can be applied to some net segments but not others. Another example is **set_target_library_subset** constraints that restrict available lib_cells in different logical hierarchies. Either of these constraints can cause one segment of a net to be unbufferable while other segments remain bufferable.

In the second use model, the **-driver** and **-loads** options are used to specify a net driver and one or more net loads which must be buffered. This approach can be used to do a more restricted bufferability query on a net than using the **-nets** option. For example, if using **-nets** shows the net has no available buffering supply, you can use **-driver** and **-loads** to remove some of the loads from consideration, and see if the net is bufferable for the remaining loads.

The **-driver** and **-loads** options can also be used to perform a hypothetical bufferability query between a certain set of related supplies. The specified driver and loads do not have to belong to the same net, or hierarchy, or voltage area. You can specify any arbitrary drivers and loads in the design. When making such a hypothetical query, you are asking "what is the bufferability of a net with the same related supply as the specified driver, going to loads with the same related supplies as the specified loads?"

When using the **-driver** and **-loads** options you must also use the **-hierarchy** option. This specifies which logical design hierarchy in which to check bufferability. This information is required to do an accurate bufferability query, because things like target library subset constraints or bias optimization have hierarchy dependent impact on bufferability.

When using **-driver, -loads,** and **-hierarchy**, the command tries to identify a real net segment in the design that corresponds to those specified options. If it finds one, then it prints an information, and can perform the full range of bufferability checks. If the command cannot identify a real net segment, then certain bufferability checks like dont_touch and is_ideal cannot be performed, since they are associated with net segments.

When using **-driver** and **-hierarchy** without **-loads** option, it specifies a special situation for a hypothetical or real dangling net without any loads.

In both use models, the voltage area is very important, as it defines the primary supply and available supplies that could be used for buffering. Without **-voltage_area** option, the command will choose native voltage area of the given net or hierarchy.

You can check bufferability either in the native voltage area of the net or in any arbitrary voltage area. Normally, buffers can only be physically placed in the voltage area that is native to the net segment being buffered. By enabling the physical feedthrough buffering feature, you can disconnect the logical from the physical and allow buffers to be placed in voltage areas non-native to the net segment hierarchy. This is enabled by:

set_app_options -name opt.common.allow_physical_feedthrough -value true

create_voltage_area_rule -allow_physical_feedthrough true

By enabling the logical feedthrough buffering feature, you expect buffers are placed in the new hierarchy. The new hierarchy is non-native for the given net segment, but native for the given voltage area. Furthermore, additional new hierarchical ports need to be punched from net segment hierarchy to new hierarchy to connect the buffers. This is enabled by:

create_voltage_area_rule -allow_logical_feedthrough true

If you use **check_bufferability** to query a net segment that exists in a logical hierarchy that is not native to the specified voltage area, this implies that you are doing a logical or physical feedthrough buffering query. **check_bufferability** issues a warning if such a query is done when both logical and physical feedthrough buffering are set to false.

=== Buffering Supply Availability ===

The first thing checked by **check_bufferability** is the supply net availability for buffering. Supply availability is determined first by finding the related supply of the driver pin and all load pins. You can manually check this using the **get_related_supply_nets** command. The set of candidate supplies for buffering is all the related supplies of the driver and loads of the net, plus their PST-equivalent supplies. This set of candidates is then filtered based on the supply availability in the specified voltage area. Finally, a buffering supply is selected from among the candidate supplies that meets these criteria:

- Does not introduce an isolation violation between the driver and any of the loads. To satisfy this condition, the chosen supply must be less-than-or-equal on compared to the driver supply, and more-than-or-equal-on compared to all load supplies
- Does not introduce a voltage shifting violation
- Single rail buffering is preferred to dual rail buffering, so the domain primary supply is chosen if possible

This process is used for determining the buffering supply of most nets. Some nets have special loads that require a different approach. These special loads include isolation cells, enable level-shifters, and overdriven level shifters. The related supply of these special cell inputs is invalid -- it is just the same as the output related supply. To guarantee that no MV violation is introduced, the buffering supply is restricted to only the driver supply for such nets. Using any other supply risks introducing an MV violation that cannot be detected until simulation is run on the output netlist from IC Compiler II.

check_bufferability reports the results of the supply availability check as a set of supplies that can be used: power, ground, as well as nwell and pwell supplies for bias designs. It also reports the type of buffer needed to use those supplies:

- Single rail: If the buffering power supply is the domain primary power supply
- Dual rail: If the buffering power supply is not the domain primary power supply
- Insulated dual rail: For bias design only, when a dual rail buffer is needed and the required backup supply for the buffer is more on than the domain primary nwell supply

As previously mentioned, which net segment is queried by **check_bufferability** is critical to the results of the buffering supply availability check. To get a complete picture of net bufferability, it is important to query a net segment from each power domain that the net logically enters. The first net segment encountered in each power domain is typically the most useful to query. It will give the most pessimistic result, since it will have the most load fanouts that could have different or even conflicting related supplies. Also, in most cases, it is most representative of what the data or CTS buffering engines will see for the available buffering supply.

By default, **check_bufferability** prints a one line summary message that gives the buffering supply information. To get more detailed information about the related supplies and special MV characteristics of any of the driver or load pins, use the **-verbose** option.

=== lib_cell Availability ===

If a valid buffering supply is determined, then **check_bufferability** next evaluates available lib_cells that can be used to buffer the net. If no valid buffering supply is determined, then lib_cell checking cannot be performed, and **check_bufferability** issues a warning.

check_bufferability evaluates every buffer and inverter lib_cell in the libraries as candidates for buffering the net. There are 7

reasons that a buffer or inverter cannot be used:

1. lib_cell purpose exclusion: The lib_cell has been excluded based on the **set_lib_cell_purpose** settings. **check_bufferability** uses an appropriate lib_cell purpose for querying buffering, depending on which engine can do high-fanout buffering for the net. It uses *optimization* purpose for data nets, *cts* purpose for clock nets, *cts* purpose for mixed clock and data nets that have been correctly isolated by the tool using clock/data isolation buffers, and both *optimization* and *cts* purpose for any mixed clock and data nets that have not been isolated from each other.
2. dont_touch constraint: If **set_dont_touch** is applied on the lib_cell, it cannot be used for buffering.
3. Target library subset exclusion: **set_target_library_subset** constraints exclude this lib_cell from being used in the hierarchy of the **-nets** or **-hierarchy** option.
4. site_def mismatch: The site_def of the lib_cell doesn't match the site_def of any site_rows in the specified voltage area. This can also occur if the site_def names match, but the characteristics are mismatched, like if the height of the lib_cell site_def and site_row site_defs is mismatched. To manually check site_def information, use the site_def attribute on lib_cell, site_row, or site_array objects.
5. Bias mismatch: This occurs for lib_cells that don't match the bias context. For example, any lib_cells without nwell and pwell pg_pins would be rejected for use in a bias block. Similarly, insulated dual-rail lib_cells would be excluded if regular dual-rail lib_cells are required.
6. Voltage mismatch: The lib_cell pg_pin voltage doesn't match that required by the buffering supply. For example, the net needs to be buffered with a 1.0V supply, but the lib_cell supply is 2.0V.
7. Incompatible supply rails: This usually occurs if the number of pg_pins on the lib_cell doesn't match what is expected. For example, if a single-rail buffer is needed, then dual-rail buffers are rejected due to the extra backup power pin.

check_bufferability reports the lib_cell availability as a count of available buffers and inverters on the same summary line that shows the buffering supply information. For example:

```
Information: Can insert single rail buffers (25 lib_cells) and inverters (33
lib_cells) with supply nets (power: VDD, ground: VSS). (MV-453)
```

To get more detailed information that lists every available buffer and inverter, as well as every unavailable buffer and inverter including the reason that it is unavailable, use the **-verbose** option.

=== Bufferability Problems ===

After performing buffering supply checking and lib_cell availability checking, **check_bufferability** also checks for buffering problems related to design constraints and other factors. Any of these that occur are listed as part of the standard output of **check_bufferability** under the 'Bufferability Problems' section. These reasons are:

- The net has no driver
- The net has multiple drivers
- The net is dont_touch
- The net is ideal
- The net is driven by constant 1 or 0
- The net is tri-state
- The net belongs to a read-only block
- The net is not in the block of the given voltage area
- The net's hierarchy is not in the domain scope of the given voltage area
- Freeze ports are found and not able to find LFT hierarchy

- No LFT buffer insertion is allowed for the given voltage area, please check its voltage area routing rule.
- Logical or physical feedthrough buffering must be enabled to buffer the net in the given voltage area
- No legal supply nets are available for buffering
- No legal supply nets are available for buffering: nwell/pwell supply nets are less on than primary power/ground for the given voltage area
- No legal lib cells are available for buffering

In some cases, all of these checks cannot be performed. **check_bufferability** always prints a warning if any checks are skipped.

The 'No legal supply nets area available for buffering' message occurs if the buffering supply checking fails. In this case, lib_cell availability checking is skipped.

The 'No legal lib cells are available for buffering' message occurs if 0 buffers and 0 inverters are available based on the lib_cell availability checking.

Checking of net dont_touch and is_ideal properties can only be done if a real net segment is identified by the command. They are skipped if the **-driver** and **-hierarchy** options are used to perform a hypothetical bufferability query.

If the net has no driver, then supply net availability and lib_cell availability checking is skipped.

Multicorner-Multimode Support

This command uses information from both active and inactive scenarios.

EXAMPLES

Check bufferability for all net segments with prefix SNPS in native voltage area.

```
prompt> check_bufferability -nets [get_nets -hierarchical {SNPS*}]
```

Check bufferability for net n1 in voltage area VA1. It is bufferable with single rail buffers.

```
prompt> check_bufferability -nets n1 -voltage_area VA1


Information: Can insert single rail buffers (8 lib_cells) and inverters (6
lib_cells) with supply nets (power: VDD1, ground: VSS). (MV-453)
```

Check bufferability for net n2 in voltage area VA2. It is bufferable with dual rail buffers.

```
prompt> check_bufferability -nets n2 -voltage_area VA2


Information: Can insert dual rail buffers (14 lib_cells) and inverters (14
lib_cells) with supply nets (power: VDD2, ground: VSS). (MV-454)
```

Check bufferability for net n3 in voltage area VA3. It is not bufferable because there are no lib_cells available for buffering. But, a valid supply for dual rail buffering is found.

```
prompt> check_bufferability -nets n3 -voltage_area VA3


Information: Cannot insert dual rail buffers or inverters because of no
```

```
    available lib_cells. But required supply nets (power: VDD3, ground: VSS) are
    available. (MV-457)
```

Check bufferability for net n4 in voltage area VA4. It has a valid buffering supply and available lib_cells, but is not bufferable for other reasons.

```
prompt> check_bufferability -nets n4 -voltage_area VA4


    Information: Cannot insert buffers or inverters because of non-MV reasons. But
    single rail buffers (12 lib_cells) and inverters (14 lib_cells) with supply
    nets (power: VDD4, ground: VSS) are available. (MV-463)
```

SEE ALSO

create_voltage_area_rule(2)
get_related_supply_nets(2)
set_dont_touch(2)
set_ideal_network(2)
set_lib_cell_purpose(2)
set_target_library_subset(2)
create_secondary_pg_placement_constraints(2)
opt.common.allow_physical_feedthrough(3)

check_bump_spacing

Check spacing among bumps.

SYNTAX

status **check_bump_spacing**
[-reference_cells *ref_cell_list*]
[-report]
[-min_spacing *distance*]
[-max_count *number*]
[-distance_style *flyline_distance* | *manhattan_total* | *manhattan_max*]

Data Types

ref_cell_list list
distance float
number integer

ARGUMENTS

-reference_cells *ref_cell_list*

Specifies the referene of bump cells to be considered during current run. If this option is not specified, all bump cells are considered.

-report

If selected, the command will report bump location statistics. No checking will be performed.

-min_spacing *distance*

Specifies the minimum allowable distance. This option must be used for bump spacing checking. This option cannot be used with *-report* option.

-max_count *number*

Specifies the maximal numebr of violations to be reported. This option cannot be used with *-report* option. The default value is 10.

-distance_style *flyline_distance* | *manhattan_total* | *manhattan_max*

Specifies how the distance is computed. If *flyline_distance* is chosen, the distance is the distance bwtween the centers of the two bumps. If *manhattan_total* is chosen, the distance is the Manhattan distance bwtween the centers of the two bumps. If *manhattan_max* is chosen, the distance is either the horizontal or vertical distance, whichever is larger, bwtween the centers of the two bumps. The default choice is *flyline_distance*.

DESCRIPTION

This command reports the spacing among bumps or violations of bump spacing. The distance between a pair of bumps is computed using the corresponding centers of the bumps.

EXAMPLES

The following command reports the bump spacing.

```
prompt> check_bump_spacing -report
```

The following command reports whether there exists pairs of bumps whose distance is less than 10 micron. At most 100 violations will be reported.

```
prompt> check_bump_spacing -min_spacing 10 -max_count 100
```

SEE ALSO

create_bump_array(2)

check_busplan_constraints

Checks the validity of the current busplan constraints defined by the **set_busplan_constraints** command.

SYNTAX

status **check_busplan_constraints**
[-ref_buses]
[-check_ref]
[buses]

Data Types

buses string

ARGUMENTS

-ref_buses

Returns a list of "referencing" busplans, based on the list of passed-in buses. If bus1 is referenced by bus2 specified with the *set_busplan_constraints -from bus2 -to bus1 -type equal* command, then **check_busplan_constraints -ref_buses bus1** will return the list containing bus2.

-check_ref

Checks the constraints of the referencing buses instead of checking the constraints of the passed-in buses.

The following command:

set result [check_busplan_constraints -check_ref $busList]

is equivalent to these commands:

set refList [check_busplan_constraints -ref_buses $busList]
set result [check_busplan_constraints $refList]

By default, the command checks the constraints of the passed-in buses.

buses

Specifies the names of busplans to check for constraint validity. The buses must be created with the **create_busplans** command.

DESCRIPTION

Checks the validity of current busplan constraints as previously defined by the **set_busplan_constraints** command. The result will be a Tcl list containing 2 Tcl sub-lists. The first sub-list contains a list of busplans that are currently failing to match their constraint. The second sub-list contains a list of busplans that will potentially fail to match their constraints, given the current suggestions from pipeline register planning.

For example, suppose bus1 has the constraint <= 3, and is included in the "buses" parameter.

If bus1 has had 4 virtual registers inserted for planning purposes, then bus1 will be included in the first sub-list, since the current situation fails the constraint (n <= 3) and the second sub-list, since it is also potentially failing in the current state.

On the other hand, if bus1 has had no virtual register inserted, but planning computations suggests bus1 needs 4 or more registers total to meet timing, bus1 will be placed into the second sub-list only. The bus currently meets its constraint, however, following the planning suggestions will cause it to fail its constraint. If bus1 can meet timing with 1, 2 or 3 virtual registers (whether inserted or computed), the constraint will be considered met and bus1 will not be included in either sub-list.

EXAMPLES

Create a constraint for a busplan named bus1, and check the constraint. Assume busplan bus1 needs 2 registers to meet timing. In this example, the 2 registers are inserted with the **modify_busplan** command.

```
prompt> set_busplan_constraints -from bus1 -to_value 2 -type equal
1
prompt> check_busplan_constraints bus1
{} {}

# Add 2 virtual registers
prompt> modify_busplan -add_register -after group1 \
  -location {100 100} bus1
{register1}
prompt> modify_busplan -add_register -after group1 \
  -location {100 100} bus1
{register2}

# If timing is met
prompt> check_busplan_constraints bus1
{} {}

# If timing is not met
prompt> check_busplan_constraints bus1
{} {bus1}

# Insert another register
prompt> modify_busplan -add_register -after group1 \
  -location {100 100} bus1
{register3}

prompt> check_busplan_constraints bus1
{bus1} {bus1}
```

SEE ALSO

create_busplans(2)
get_busplans(2)
load_busplans(2)
modify_busplan(2)
report_busplan_constraints(2)
report_busplans(2)
set_busplan_constraints(2)

check_clock_gate_library_cell_availability

Checks whether there are ICG library cells available for clock gate insertion in the design.

SYNTAX

status **check_clock_gate_library_cell_availability**
[-objects *object_list*]
[-verbose]

Data Types

object_list list

ARGUMENTS

-objects *object_list*

Specifies the objects on which the check will apply. Valid objects are hierarchical instances, power domains and designs. The check will be performed on all the hierarchical instances assciated with the objects in *object_list*. If the option is not used, the check will be performed on all the hierarchical instances in the current design.

-verbose

Shows more information for every hierarchical instance on which the check failed.

DESCRIPTION

This command will check whether there are ICG library cells available, according with the current library constraints and clock gate style settings, for every hierarhical instance indicated in parameter *-objects*, or in the entire design.

The command will succeed (and return 1) if all the checked hierarchical cells have ICG library cells available for clock gate insertion.

The command will fail (and return 0) if at least one hierarchical cell exists with no ICG library cells available for insertion.

ICG library cells in the library can be rejected because of the following reasons:

\(bu General Reasons, which apply to the entire design. +0.25i

- [LIBRARY]

  \n+[step]. "it is a dont_touch libcell or it does not match the expected libcell purpose".

\(bu Instance-specific Reasons, which depends on the hierarchical cell. +0.25i

- [STYLE]

  \n+[step]. "it has no te port when required"
  \n+[step]. "it has no obs port when required"
  \n+[step]. "its en port does not match active high/low requirements => it is active high, when active low was expected" or vice versa.
  \n+[step]. "its te port does not match active high/low requirements => it is active high, when active low was expected" or vice versa.
  \n+[step]. "its clk output port does not match inversion requirements => it is inverted, when non-inverted was expected" or vice versa.
  \n+[step]. "its obs port does not match inversion requirements => it is inverted, when non-inverted was expected" or vice versa.
  \n+[step]. "it does not match sequential/latch requirements => it is latch-based, when latch-free was expected" or vice versa.
  \n+[step]. "it does not match edge requirements => it is rising edge, when falling edge was expected" or vice versa.
  \n+[step]. "it does not match pre/post control requirements => it is pre-control, when post-control was expected" or vice versa.

- [OPCONDS]

  \n+[step]. "it does not match hierarchy operating conditions".

EXAMPLES

The following example shows the output in case of success on all the hierarchies in the design.

```
prompt> check_clock_gate_library_cell_availability
There are ICG library cells available on all hierarchies. (CGT-4303)
1
```

The following example shows the output in case of success on objects mid1 and mid2.

```
prompt> check_clock_gate_library_cell_availability \
  -objects [get_cells {mid1 mid2}]
There are ICG library cells available on all objects. (CGT-4303)
1
```

This example shows the output in case of failure when there are no ICG library cells in the libraries.

```
prompt> check_clock_gate_library_cell_availability
Warning: There are no ICG library cells available in the library. (CGT-3301)
0
```

This example shows the output in case of failure when there are no ICG library cells for instance mid1 and mid2.

```
prompt> check_clock_gate_library_cell_availability \
  -objects [get_cells {mid1 mid2 mid3}]
Warning: There are no ICG library cells available for hierarchical cell
  mid1. (CGT-3302)
Warning: There are no ICG library cells available for hierarchical cell
  mid2. (CGT-3302)
0
```

This example shows the verbose output in case of failure when there are no ICG library cells for instance mid1 and mid2, after setting a clock gate style that requires observation output, when some of the library cells are restricted and the remaining do not have observation output port.

```
prompt> set_clock_gate_style -observation_output \
  -target {neg_edge_flip_flop pos_edge_flip_flop} \
  -objects [get_cells {mid1 mid2 mid3}]
1
prompt> check_clock_gate_library_cell_availability \
  -objects [get_cells {mid1 mid2 mid3}] \
  -verbose
Some hierarchies have no ICG libcells available. (CGT-4304).
ICG library cell tcbn90ghvtbc/CKLHQHVTD1 cannot be used for all-designs
  because it is a dont_touch libcell or it does not match the expected
  libcell purpose. (CGT-4305)
ICG library cell tcbn90ghvtbc/CKLHQHVTD2 cannot be used for all-designs
  because it is a dont_touch libcell or it does not match the expected
  libcell purpose. (CGT-4305)
ICG library cell tcbn90ghvtbc/CKLHQHVTD3 cannot be used for all-designs
  because it is a dont_touch libcell or it does not match the expected
  libcell purpose. (CGT-4305)
Warning: There are no ICG library cells available for hierarchical cell
  mid1. (CGT-3302)
ICG library cell tcbn90ghvtbc/CKLHQHVTD4 cannot be used for mid1 because
  it has no obs port when required. (CGT-4305)
ICG library cell tcbn90ghvtbc/CKLHQHVTD6 cannot be used for mid1 because
  it has no obs port when required. (CGT-4305)
ICG library cell tcbn90ghvtbc/CKLHQHVTD8 cannot be used for mid1 because
  it has no obs port when required. (CGT-4305)
Warning: There are no ICG library cells available for hierarchical cell
  mid2. (CGT-3302)
ICG library cell tcbn90ghvtbc/CKLHQHVTD4 cannot be used for mid2 because
  it has no obs port when required. (CGT-4305)
ICG library cell tcbn90ghvtbc/CKLHQHVTD6 cannot be used for mid2 because
  it has no obs port when required. (CGT-4305)
ICG library cell tcbn90ghvtbc/CKLHQHVTD8 cannot be used for mid2 because
  it has no obs port when required. (CGT-4305)
0
```

SEE ALSO

set_clock_gate_style(2)

check_clock_trees

Checks the clock trees of current design for possible problems with netlist, timing constraints, clock constraints, routing constraints, or other tool configurations that can adversely impact clock tree synthesis.

SYNTAX

status **check_clock_trees**
[-clocks clock_list]
[-message_limit *number*]

Data Types

object_list list or collection
number integer

ARGUMENTS

-clocks clock_list

Checks only those clock trees that appear in the clock_list. By default, the command checks all clocks in the design.

-message_limit *number*

Specifies the number of messages to be printed in each checks detail report. Default all of the messages will be printed. Number should be greater than 0

DESCRIPTION

Use the check_clock_trees command before clock tree synthesis to check for common problems that might impact clock tree synthesis.

EXAMPLES

SEE ALSO

check_consistency_settings

Compares the settings between this tool and the PrimeTime, StarRC, or IC Compiler tool for good timing correlation between the two tools.

SYNTAX

status **check_consistency_settings**
[-tool *type*]
[-work_dir *path_name*]
[-output *filename*]
[-script *filename*]

Data Types

type string
path_name string
filename string

ARGUMENTS

-tool

Indicates the target tool to check against. Valid values are pt, icc, icc2, and starrc.

-work_dir

Specifies the directory that contains the scripts to check. If the script file specified by the **set_consistency_settings_options** command includes some nested scripts, you must use this option to ensure those nested scripts can be sourced correctly.

-output

Specifies the output file to use to save the correlation checking result.

-script

Specifies the output file to use to save the tcl file which make ICC2 RC extraction setting the same as StarRC.

DESCRIPTION

The **check_consistency_settings** command compares the settings between this tool and the PrimeTime, StarRC, or IC Compiler tool for good timing correlation between the two tools. The command reports an error for each setting that is different from the recommended setting and reports the proper setting in the applicable tool.

Before using this command, run the **set_consistency_settings_options** command to set up the executable location and setup script.

Multicorner-Multimode Support

This command applies only to current scenario for timing correlation checking, or applies to the **-corner** and **-early** options of **set_consistency_settings_options** command for extraction correlation checking.

EXAMPLES

The following example uses the **check_consistency_settings** command to check consistency between this tool and the PrimeTime tool. The **set_consistency_settings_options** command sets up the executable location and related setup script such as pt.tcl.

```
prompt> set_consistency_settings_options \
  -exec_path /software/syn/bin -tool pt -script pt.tcl
prompt> check_consistency_settings -tool pt
Error: The timing_remove_clock_reconvergence_pessimism setting is
 inconsistent between Fusion Compiler and PrimeTime. The Fusion Compiler
 value is TRUE; the PrimeTime value is "false". (CORR-803)
Error: The timing_remove_clock_reconvergence_pessimism PrimeTime variable
should be set to TRUE. (CORR-800)
```

The following example uses the **check_consistency_settings** command to check consistency between the tool and the StarRC tool. The **set_consistency_settings_options** command sets up the executable location and related setup script such as starrc.tcl.

```
prompt> set_consistency_settings_options \
  -exec_path /software/syn/bin -tool starrc -script starrc.tcl \
  -corner maxCorner
prompt> check_consistency_settings -tool starrc
Error: The temperature(OPERATING_TEMPERATURE) setting is inconsistent
  between Fusion Compiler and StarRC. The Fusion Compiler value is 25;
  the StarRC value is 0.00. (CORR-803)
Error: The virtual_shield_extraction Fusion Compiler setting should be set
  to FALSE. (CORR-812)
Error: The TEMPERATURE_SENSITIVITY StarRC setting should be set to NO.
  (CORR-800)
```

SEE ALSO

set_consistency_settings_options(2)

check_design

Runs pre-defined or user-defined checks on current design. Messages generated by the checks run, are captured into an EMS database file and also into a text log file. Summary of the messages generated is also reported.

SYNTAX

status **check_design**
-checks *names_of_checks*
[-ems_database *ems_database_name*]
[-open_message_browser]

Data Types

names_of_checks list
ems_database_name string

ARGUMENTS

-checks *names_of_checks*

List of the names, of one or more user-defined or pre-defined checks, which will run on the current design.

A check pre-defined by the tool is called a pre-defined check. Examples of pre-defined checks are **pre_route_stage**, **mv_design**, **clock_trees**, **hier_timing**, **hier_pre_placement** etc. An user can create her own check with **create_check_design_strategy** command, which are called user-defined checks. User can get the list of all pre-defined and user-defined checks by invoking **get_design_checks** command.

Check **mv_design** is called an atomic-check since it is defined using a single UI command **check_mv_design**. Check **pre_route_stage** is called a mega-check since it is defined as a collection of atomic-checks.

Checks **pre_placement_stage**, **pre_clock_tree_stage** and **pre_route_stage** are the three available pre-defined mega-checks. Check **pre_placement_stage** can be run on a design in pre-placement stage. Similarly, checks **pre_clock_tree_stage** and **pre_route_stage** can be run on a design in pre-clock-tree stage and pre-route stage respectively.

There are several pre-defined atomic-checks like **mv_design**, **routability**, **clock_trees**, **hier_timing**, **hier_pre_placement** etc., which can be run by the user based on her requirements at different stages of design.

Below is a list of pre-defined atomic checks and mega-checks. You can run get_design_checks command to get this list as well.

Atomic-checks: cts_qor, design_extraction, legality, rp_constraints, feedthroughs, mib_alignment, finfet_grid, pg_drc, pg_missing_vias, pin_placement, routes, physical_constraints, dp_pre_floorplan, dp_pre_create_placement_abstract, dp_pre_block_shaping, dp_pre_macro_placement, dp_pre_power_insertion, dp_pre_pin_placement, dp_pre_push_down, dp_pre_create_timing_abstract, dp_pre_timing_estimation, dp_pre_budgeting, dp_pre_clock_trunk_planning, dp_floorplan_rules, mv_design, routability, clock_trees, scan_chain, design_mismatch, analyze_design_violations, timing, hier_timing, hier_pre_placement, hier_pre_clock_tree, hier_pre_route, hier_pre_compile, block_ready_for_top, netlist,

3d_netlist, unbound, design_states.

Mega-checks: pre_placement_stage, pre_clock_tree_stage, pre_route_stage.

-ems_database *ems_database_name*

Name of EMS database which will store the Error, Warning and Info messages generated by the checks.

If this option is not specified, then by default, "check_design.ems" is used as the name of EMS database and if an EMS database by name "check_design.ems" already exists, then its contents will be overwritten.

The EMS database name can contain only alpha-numeric characters and underscore character("_"). User is expected to provide the EMS database name with suffix ".ems". If user does not provide the suffix ".ems", then suffix ".ems" will be appended to the given name, automatically by the command, and used as EMS database name.

-open_message_browser

Opens the message browser window in GUI, which displays the messages captured in the EMS database.

DESCRIPTION

This command is an unified framework to perform design checks. The command writes the Error, Warning and Info messages generated by running checks on design, to Enhanced Messaging System(EMS) database file and into a text log file. All the messages generated by the checks are captured into text log file. Only a sub-set of messages generated are captured into EMS database. The messages NOT captured into EMS database are referred to as Non-EMS messages. Summary information is displayed by the command to indicate the number of EMS and Non-EMS messages along with their message-ids and message templates.

To look into the Error, Warning and Info messages generated by the checks and debug the issue, one can open the EMS database in GUI, by opening message browser window. If user does not prefer GUI then the messages can be viewed in text log file. The name of the log file is time-stamp based and a new log file is generated for every run of **check_design** command. The log file name is of the form "check_design<T>.log", where "<T>" indicates the time-stamp. The log file is saved in the current directory.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information. But checks run by the command may depend on scenario-specific information.

EXAMPLES

The following example runs the predefined mv_design atomic check.

```
prompt> check_design -checks mv_design

****************************************
 Report : check_design
 Options: { mv_design }
 Design :
 Version:
 Date   : Tue May 19 03:28:59 2015
****************************************
```

```
Running atomic-check "mv_design"

 *** EMS Message summary ***
 ------------------------------------------------------------------------------
 Rule   Type   Count Message
 ------------------------------------------------------------------------------
 MV-008 Error  4     PG net "%pgNet" is not defined in UPF as a supply net.
 MV-050 Error  343   %type related supply net(s) of signal pin "%signalPin" cannot be...
 MV-027 Warn   23    The tie-off connection '%tieNet' has not been implemented.
 ------------------------------------------------------------------------------
 Total 370 EMS messages : 347 errors, 23 warnings, 0 info.
 ------------------------------------------------------------------------------

 *** Non-EMS message summary ***
 ------------------------------------------------------------------------------
 Total 0 non-EMS messages : 0 errors, 0 warnings, 0 info.
 ------------------------------------------------------------------------------

Info: EMS database is saved to file "check_design.ems".
prompt>
```

The following example runs the predefined pre_placement_stage mega check.

```
prompt> check_design -checks pre_placement_stage
 ****************************************
  Report : check_design
  Options: { pre_placement_stage }
  Design :
  Version:
  Date   : Tue May 19 03:31:28 2015
 ****************************************

 Running mega-check "pre_placement_stage":
    Running atomic-check "design_mismatch"
    Running atomic-check "scan_chain"
    Running atomic-check "mv_design"
    Running atomic-check "rp_constraints"
    Running atomic-check "timing"

 *** EMS Message summary ***
 ------------------------------------------------------------------------------
 Rule    Type   Count Message
 ------------------------------------------------------------------------------
 MV-008  Error  4     PG net "%pgNet" is not defined in UPF as a supply net.
 MV-050  Error  343   %type related supply net(s) of signal pin "%signalPin" cannot be...
 TCK-011 Warn   4     A loop "%loop" is detected, the arc "%arc" is disabled to break ...
 TCK-002 Warn   6090  The register clock pin "%pin" has no fanin clocks.
 MV-027  Warn   23    The tie-off connection "%tieNet" has not been implemented.
 TCK-001 Warn   19193 The reported endpoint "%endpoint" is unconstrained. Reason: "%re...
 DFT-011 Info   1     The design has no scan chain defined in the scandef.
 DMM-107 Info   1     Mismatch-type %mmtype is detected on object-type %objtype for o...
 ------------------------------------------------------------------------------
 Total 25659 EMS messages : 347 errors, 25310 warnings, 2 info.
 ------------------------------------------------------------------------------

 *** Non-EMS message summary ***
```

```
    ----------------------------------------------------------------------------
    Rule     Type   Count Message
    ----------------------------------------------------------------------------
    NDMUI-173 Error  1     There are no relative placement groups in the design.
    PVT-032   Info   1     Corner %s: no PVT mismatches.
    ----------------------------------------------------------------------------
    Total 2 non-EMS messages : 1 errors, 0 warnings, 1 info.
    ----------------------------------------------------------------------------

  Info: EMS database is saved to file "check_design.ems".
  Info: Non-EMS messages are saved to file "check_design2015May19033128.log".
prompt>
```

The following example runs the predefined pre_clock_tree_stage mega check.

```
prompt> check_design -checks pre_clock_tree_stage
```

The following example runs the predefined pre_route_stage mega check.

```
prompt> check_design -checks pre_route_stage
```

The following example runs the pre_route_stage, mv_design, and routability checks.

```
prompt> check_design -checks {pre_route_stage mv_design routability}
```

The following example creates a user-defined atomic-check named my_atomic_check and runs it.

```
prompt> create_check_design_strategy -define_check my_atomic_check \
  {check_mv_design -power_connectivity -max_message_count 0}
prompt> check_design -checks my_atomic_check
```

The following example creates a user-defined mega-check named my_mega_check and runs it.

```
prompt> create_check_design_strategy -define_group my_mega_check \
  {my_atomic_check mv_design routability}
prompt> check_design -checks my_mega_check
```

The following example runs the mv_design check and stores the output messages in an EMS database file named my_database.ems.

```
prompt> check_design -checks mv_design -ems_database my_database.ems
```

The following example runs the mv_design check and opens the EMS database in the GUI.

```
prompt> check_design -checks mv_design -open_message_browser
```

The following example checks for issues related to design planning before floorplanning.

```
prompt> check_design -checks { dp_pre_floorplan }
```

The following example checks for issues related to design planning before creating placement abstracts and shaping blocks.

```
prompt> check_design \
  -checks { dp_pre_create_placement_abstract dp_pre_block_shaping }
```

SEE ALSO

create_check_design_strategy(2)
report_check_design_strategy(2)
get_design_checks(2)

check_design_for_clock_trunk_planning

Check the design for CTP feasibility.

SYNTAX

status **check_design_for_clock_trunk_planning**
[-clocks *clock_list*]
[-summary]

Data Types

clock_list collection

ARGUMENTS

-clocks *clock_list*

Specifies the clocks for which clock trunk planning checker is to be run. By default, all clocks are checked.

-summary

This boolean argument specifies if the summary report is to be printed.

DESCRIPTION

This command runs the CTP related checks from top.

EXAMPLES

The following example runs CTP checker for clock 'clk1' and also prints the summary.

prompt> **check_design_for_clock_trunk_planning -clocks clk1 -summary**

SEE ALSO

check_hier_design(2)

check_design_states

Checks design states of current block like linked, mapped.

SYNTAX

status **check_design_states**
[-verbose]

ARGUMENTS

-verbose

Prints the unmapped cells information.

DESCRIPTION

This command checks the linked and mapped design states of the current design. This command will print whether the design is linked or not and the mapped percentage of the cells of the current design.

When command option **-verbose** is specified the command will report the unmapped cells information.

This command has no dependency on scenario-specific information.

EXAMPLES

The check_design_states command displays the following information when no option is specified.

```
prompt> check_design_states
****************************************
Report : check_design_states
Design : top
Version:
Date   : Thu Nov 28 02:17:49 2019
****************************************

Design is linked
Design is 66.67% mapped
1
```

The check_design_states command displays the following information when option '-verbose' is specified.

```
prompt> check_design_states -verbose
****************************************
Report : check_design_states
Design : top
Version:
Date   : Thu Nov 28 02:23:40 2019
****************************************

Design is linked
Design is 66.67% mapped

Information: cell reg_p_i1/I_0 is unmapped. (NDMUI-915)
Information: cell I_0 is unmapped. (NDMUI-915)
1
```

check_duplicates

Checks, reports and optionally removes any duplicate shapes and vias detected in the block.

SYNTAX

status **check_duplicates**
[-blocks {*block*}]
[-verbose]
[-remove]
[-object_types {*type_list*}]
[-return_as_collection]

Data Types

block collection
type_list list

ARGUMENTS

-blocks {*block*}

Specifies the block for finding duplicates. If this is not specified, duplicate objects will be found in the current block. A summary of the total count of duplicate objects found will be printed.

-verbose

When you specify this option, the command writes all the duplicate objects in a text file duplicates.txt.gz in the current folder. This file will contain the list of all duplicate objects found in the design. If there are no duplicates found, an empty file will be generated.

-remove

When you specify this option, the command reports and also removes all the duplicate objects found. It will generate a text file duplicates.txt.gz in the current folder which will contain detailed information about each of the duplicate objects deleted. A summary of the removed objects will be printed in the log file. If there are no duplicates found, an empty file will be generated.

-object_types *type_list*

Specifies the list of object types to be reported as duplicates. Valid values for this option are *shapes* and *vias*.

-return_as_collection

When you specify this option, this command will return duplicates in a collection. This option cannot be specified with **-remove** option.

DESCRIPTION

Checks, reports and optionally removes all the duplicates found in the specified block. If no block is specified through **-blocks** option, current block will be used. If **-verbose** option is not used, only a limited entries of duplicates will be printed in the log file. A total of 20 rows and 5 duplicate objects in each row will be printed by default. To get complete list of objects, use **-verbose** option. When this option will be used, the complete list of objects will be printed in the text file duplicates.txt.gz file in current folder.

If you specify the **-remove** option, reported duplicate objects will be removed. In the log file, a summary of total objects removed will be printed. A complete list of removed objects will be printed in the text file duplicates.txt.gz file created in current folder.

EXAMPLES

The following example reports duplicate objects found in the current block.

```
prompt> check_duplicates
****************************************
Report : Reporting Duplicates
Design : ORCA
Version:
Date   : Mon Mar 20 23:50:57 2017
****************************************
Block   Layer    Type      Objects
-------------------------------------------------------
ORCA    METAL    path      PATH_14_24338  PATH_14_24340
1

***Summary of duplicates***
Number of duplicate shapes found = 1
1
```

The following example checks and reports duplicate objects in specified block.

```
prompt> check_duplicates -blocks orca_hier_lib:ORCA.design
****************************************
Report : Reporting Duplicates
Design : ORCA
Version: M-2016.12-SP4-BETA
Date   : Tue Mar 21 01:36:45 2017
****************************************
Block     Layer    Type      Objects
-----------------------------------------------------------------------
ORCA      METAL2   rect      RECT_18_0  RECT_18_2508
ORCA      METAL2   rect      RECT_18_2504  RECT_18_2505  RECT_18_2506
ORCA      METAL2   polygon   POLYGON_18_2502  POLYGON_18_2507
ORCA      METAL3   rect      RECT_28_1  RECT_28_1739
ORCA      METAL3   rect      RECT_28_1741  RECT_28_1742  RECT_28_1743

***Summary of duplicates***
Number of duplicate shapes found = 7
1
```

The following example checks, reports and removes duplicate objects from the current block.

```
prompt> check_duplicates -remove
****************************************
Report : Reporting Duplicates
Design : test
Version:
Date   : Tue Mar 21 01:41:22 2017
****************************************
Block     Layer    Type      Objects
--------------------------------------------------------------------------
test      METAL2   rect      RECT_18_0  RECT_18_2501
test      METAL2   rect      RECT_18_2504  RECT_18_2505  RECT_18_2506
test      METAL2   polygon   POLYGON_18_2502  POLYGON_18_2507
test      METAL3   rect      RECT_28_1  RECT_28_1739
test      METAL3   rect      RECT_28_1741  RECT_28_1742  RECT_28_1743

***Summary of duplicates***
Number of duplicate shapes found = 7.
Number of duplicate shapes removed = 7.

Please refer to file /usr/duplicates.txt.gz for complete list of all duplicates.
1
```

The following example shows usage of verbose option.

```
prompt> check_duplicates  -verbose
****************************************
Report : Reporting Duplicates
Design : test
Version:
Date   : Mon Apr 17 02:00:16 2017
****************************************
Block     Layer    Type      Objects
--------------------------------------------------------------------------
test      METAL2   rect      RECT_18_0  RECT_18_2501
test      METAL2   rect      RECT_18_2504  RECT_18_2505  RECT_18_2506
test      METAL2   polygon   POLYGON_18_2502  POLYGON_18_2507
test      METAL3   rect      RECT_28_1  RECT_28_1739
test      METAL3   rect      RECT_28_1741  RECT_28_1742  RECT_28_1743

***Summary of duplicates***
Number of duplicate shapes found = 7.

Please refer to file /usr/duplicates.txt.gz for complete list of all duplicates.
1
```

The following example shows usage of the **-return_as_collection** option.

```
prompt> check_duplicates -return_as_collection
****************************************
Report : Reporting Duplicates
Design : test
Version:
Date   : Wed Apr 26 02:27:18 2017
****************************************
Block     Layer    Type      Objects
--------------------------------------------------------------------------
test      METAL2   rect      RECT_18_0  RECT_18_2501
```

```
test     METAL2   rect     RECT_18_2504  RECT_18_2505  RECT_18_2506
test     METAL2   polygon  POLYGON_18_2502  POLYGON_18_2507
test     METAL3   rect     RECT_28_1  RECT_28_1739
test     METAL3   rect     RECT_28_1741  RECT_28_1742  RECT_28_1743

***Summary of duplicates***
Number of duplicate shapes found = 7.
{RECT_18_2501 RECT_18_2505 RECT_18_2506 POLYGON_18_2507 RECT_28_1739 RECT_28_1742 RECT_28_1743}
```

The following example shows usage of object_types option.

```
prompt> check_duplicates -object_types shapes
****************************************
Report : Reporting Duplicates
Design : r4000
Version:
Date   : Thu Apr 27 01:11:20 2017
****************************************
Block     Layer    Type      Objects
--------------------------------------------------------------------------
r4000     M1       line      LINE_31_1737  LINE_31_1739
r4000     M2       line      LINE_32_3525  LINE_32_4106

***Summary of duplicates***
Number of duplicate shapes found = 2.
1
```

SEE ALSO

check_equivalent_power_domains

Check whether the power domains are equivalent and satisfy the criteria to share a single voltage area.

SYNTAX

status **check_equivalent_power_domains**
power_domains
[-functional_equivalence]
[-verbose]

Data Types

power_domains list

ARGUMENTS

power_domains

Specifies the power domains to be checked. The command fails if the number of power domains are less than two.

-functional_equivalence

Specifies that domain supplies are considered as equivalent based on functional equivalence. If this options is not specified, by default, domain supplies are considered as equivalent based on electrical equivalence.

-verbose

Shows the groups of equivalent power domains to share a primary voltage area.

DESCRIPTION

The **check_equivalent_power_domains** command checks whether the specified power domains are equivalent or not to share a voltage area. The command will return 1 if the specified power domains are equivalent and satisfy the criteria to share a primary voltage area, 0 otherwise.

This command supports the shared voltage area flow only with mv.upf.multi_pd_shared_voltage_area_flow set to new.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example compares two power domains that are not equivalent to share a voltage area.

```
prompt> check_equivalent_power_domains {pd1 pd2}
 0
```

The following example compares two power domains that are equivalent to share a voltage area.

```
prompt> check_equivalent_power_domains {pd3 pd4} -functional
 1
```

The following example compares power domains that are not equivalent to share a voltage area and shows the groups of distinct power domain subsets.

```
prompt> check_equivalent_power_domains {pd1 pd2 pd3 pd4} -verbose
Information: List of equivalent power domains - {pd1} {pd2} {pd3 pd4}. (UPF-483)
 0
```

SEE ALSO

report_power_domains(2)
create_voltage_area(2)
set_voltage_area(2)
get_equivalent_power_domains(2)

check_error

Returns true if message IDs specified in shell.common.check_error_list were encountered in previous commands.

SYNTAX

status **check_error**
[-verbose]
[-reset]

ARGUMENTS

-verbose

Displays the list of message IDs found during previous commands.

By default, the command only returns the error status.

-reset

Resets the list of previously found message IDs. After resetting the error list, the **check_error -verbose** command returns an empty list.

By default, the command does not modify the message list.

DESCRIPTION

The **check_error** command is used to compare error, warning, or informational messages issued by previous commands to the list of message IDs specified by **shell.common.check_error_list** application option. While it is most commonly used for error messages, the application option can contain error, warning, or informational message IDs.

If the **check_error** command finds message IDs that are specified in the **shell.common.check_error_list** application option, the command returns 1. If the **check_error** command does not find any messages that are specified by the **shell.common.check_error_list** application option, the command returns 0.

Messages that are suppressed by the **suppress_message** command are reported by **check_error**. If you want suppressed messages to be ignored by **check_error** then you must remove their message IDs from the **shell.common.check_error_list** application option.

EXAMPLES

The following example shows how to use the **check_error** command to check if execution errors are specified by the **shell.common.check_error_list** application option:

```
prompt> set_app_option -name shell.common.check_error_list -value {DES-001}

prompt> check_error
0

prompt> get_cells
Error: Current block is not defined. (DES-001)

prompt> check_error
1

prompt> check_error -verbose
{DES-001}
1
```

The following command resets the error list:

```
prompt> check_error -reset -verbose
{DES-001}
1

prompt> check_error -verbose
0
```

To use the **check_error** command in a script to see if specified errors occurred in the previous commands, use the following command:

```
prompt> if { [check_error] } {
   echo failure_message
   exit
} else {
   echo success_message
}
```

SEE ALSO

shell.common.check_error_list(3)
suppress_message(2)

check_feedthroughs

Checks the feedthroughs on physical blocks.

SYNTAX

string **check_feedthroughs**
[-unused_feedthroughs]
[-reused_feedthroughs]
[-redundant]
[-net_constraints]
[-topo_constraints]
[-include_original_feedthroughs]
[-include_buffered]
[-shape L | M | N]
[-pure]
[-self]
[-mixed]
[-close_to_edge]
[-tolerance_distance *feedthrough_tolerance_distance*]
[-filter_by_length]
[-feedthrough_length *feedthrough_length*]
[-cells *list_of_blocks*]

Data Types

feedthrough_tolerance_distance	float
feedthrough_length	float
list_of_blocks	list

ARGUMENTS

-unused_feedthroughs

Checks and reports feedthroughs pins that are not connected to a top-level net. Feedthroughs are not allowed in some designs. An example is a design with multiple instantiated blocks (MIB), where a feedthrough needs to be created on one MIB instance while it cannot be used by other MIB instances.

-reused_feedthroughs

Checks and reports feedthrough pins that are reused across multiple instantiated block (MIB) instances. This check only applies to MIB blocks.

-redundant

This option directs the command to look for redundant connections created on feedthrough nets. The pins it returns and reports

are unnecessarily created either in the original netlist (if the **-include_original_feedthroughs** option is used) or by pin-placement during feedthrough creation. The option cannot be used with any other option except for the **-include_original_feedthroughs** option.

-net_constraints

Checks and reports feedthrough per-net, per-block feedthrough constraint violations. The command reports nets that violate feedthrough constraints and returns the associated feedthrough pins.

-topo_constraints

Checks and reports nets that violate their associated topological constraints. The command reports nets that violate feedthrough constraints and returns the associated feedthrough pins where applicable.

-include_original_feedthroughs

Includes original feedthrough pins (not inserted by the tool) for checking. An original feedthrough is a feedthrough that exists in the original netlist. By default, original feedthrough pins are not checked.

-include_buffered

Skips over repeaters and buffers, when timing libraries are loaded, (effectively including buffered feedthroughs) when identifying, reporting, and checking feedthroughs.

-shape L | M | N

Reports feedthroughs of the specified shape. "L shaped" feedthroughs are two pin feedthroughs that have pins on adjacent edges. The **-tolerance_distance** option is used to specify how far these two feedthrough pins can be from the shared corner. "M shaped" feedthroughs have multiple-fanout pins. "N shaped" feedthroughs have feedthrough ports on the same edge. This option can only be used with the **-cells**, **-self**, or the **-tolerance_distance** options. The command also includes all feedthroughs for shape checking, including original feedthrough ports. However, "L" and "N" shape feedthrough reporting is limited to pure feedthroughs.

-pure

Reports and returns only the pure feedthroughs in the design. Pure feedthroughs are defined as feedthroughs that have no physical pins or connections (except for inverters/buffers) inside the block that they pass through. This option can only be used with the **-cells** or **-self** option.

-self

Reports and returns those ports and terminals of the top-level block that are feedthroughs. That is they are connected (potentially through multiple nets/hierarchies) to other top-level ports. This option can only be used with the **-include_original_feedthroughs**, **-include_buffered**, **-pure**, **-mixed**, **-shape**, **-close_to_edge**, and **-filter_by_length** options.

-mixed

Reports and returns only the mixed feedthroughs in the design. There are two types of mixed feedthroughs: 1. the feedthrough net has physical connections to combinational cell (besides inverters/buffers) inside the block that it passes through; 2. the feedthrough net owns both original and non-original feedthrough ports. This option can only be used with the **-cells** and **-self** options.

-close_to_edge

Checks for feedthrough ports that are within the specified distance from the same edge of the physical block. For a feedthrough port to be considered close to a given edge it should lie on a different edge and also on an edge that is of a different orientation than the edge it is close to. The check is limited to pure feedthroughs. The **-tolerance_distance** option can be used to specify the distance (in microns) from the edge. The default tolerance distance is 100 microns.

-tolerance_distance *feedthrough_tolerance_distance*

Specifies the tolerance distance that is used with the **-shape** and **-close_to_edge** options. When used with **-close_to_edge**, the feedthrough ports that are less than specified distance (in microns) from the same edge will be reported. When used with **-shape**, the distance applies to L-shaped feedthroughs only. The L-shaped feedthroughs that have both ports within specified distance from the same corner are reported. By default, the tolerance distance is 100 microns.

-filter_by_length

Checks for feedthroughs with nets whose length is less than the value specified by the **-feedthrough_length** option. Use this option to identify shorted feedthroughs. Note that the unplaced pins will be excluded in net length measurement.

-feedthrough_length ***feedthrough_length***

Reports feedthroughs for nets with a length that is less than the specified *feedthrough_length* (in microns). The option can only be used with the **-filter_by_length** option and **-self** options. If routing is present for the net, the length of the route is used. Otherwise, half perimeter wire-length (from the bounding box of the net) is used to calculate the length. By default, the feedthrough length is 1000 microns.

-cells ***list_of_blocks***

Specifies a list of blocks or cells for feedthrough checking.

DESCRIPTION

This command is used to check and report feedthroughs issues, and returns a collection of feedthrough pins that correspond to the specified arguments/filters. If no option is specified, the command simply returns a collection of all the feedthrough pins. To include original feedthrough ports (not inserted by the tool) the **-include_original_feedthroughs** option can be used. To include buffered feedthrough nets for checking/reporting the **-include_buffered** option can be used.

By default the command looks for and checks feedthroughs at multiple levels of physical hierarchy (MPH support). The **set_editability** command can be used to enable or disable checking on individual reference-blocks or levels of physical hierarchy. For net constraint and topological checking a particular net is checked only if the instance in which it originates (its parent instance/hierarchy) is enabled for design planning.

EXAMPLES

The following example checks for unused feedthrough pins on all blocks.

```
prompt> check_feedthroughs -unused_feedthroughs
```

The following example checks for L-shaped feedthroughs on all blocks.

```
prompt> check_feedthroughs -shape L
```

The following command returns all tool inserted feedthrough ports

```
prompt> check_feedthroughs
```

The following command returns all feedthrough ports including the original feedthrough ports in the design.

```
prompt> check_feedthroughs -include_original_feedthroughs
```

SEE ALSO

place_pins(2)
report_feedthroughs(2)
set_editability(2)

check_finfet_grid

Checks for violations of object placement or boundary to the FinFET grid.

SYNTAX

status **check_finfet_grid**
[-objects *object_name_or_collection*]
[-hierarchical]

Data Types

object_name_or_collection collection

ARGUMENTS

-objects *object_name_or_collection*

Specifies the objects to be checked. Currently supported object types are core_area, site_row, site_array, macro cell, I/O pad cell, flip chip river cell, and block cell. Other object types are not supported by this checker command. Specify the objects either by using an object access command, such as **get_site_rows**, or by specifying object name patterns in a Tcl list. By default, all supported objects in the current design are checked.

-hierarchical

Controls whether the objects within sub-blocks are checked. The **-hierarchical** option and **-objects** options are mutually exclusive. By default, the command only check objects in the current block.

DESCRIPTION

FinFET technology uses a grid which is about 40nm in pitch. The grid might have different pitch values in the x- and y-directions. Since FinFET is transistor in polysilicon layer, the boundary or placement of physical objects based on transistor should conform to FinFET grid. This command checks the physical design for violations against the FinFET grid. Standard cells must conform to FinFET grid. Since standard cells are placed in site rows, this command checks site rows rather than standard cells. If site rows are on FinFET grid, standard cells should be on FinFET grid automatically.

If you receive error messages when you run this command, see the man pages for the error message IDs for information on how to fix the FinFET grid violations. After you fix the violations, run this command again to verify your fixes.

EXAMPLES

The following example shows the default command run.

prompt> **check_finfet_grid**

The following example shows command run on hierarchical design.

prompt> **check_finfet_grid -hierarchical**

The following example only checks hard macro cells.

prompt> **check_finfet_grid -objects [get_cells -hierarchical -filter "design_type==macro"]**

The following example only checks I/O pad cells.

prompt> **check_finfet_grid -objects [get_cells -filter "design_type==pad" -hierarchical]**

SEE ALSO

get_cells(2)
get_site_rows(2)

check_floorplan_rules

Reports violations of related floorplan spacing, enclosure, halo, length, area, width rules and track constraints.

SYNTAX

collection **check_floorplan_rules**
[*list_of_rule_names*]
[-object_types *list_of_object_types*]
[-lib_cells *collection_of_lib_cells*]
[-exclude_rules *list_of_rule_names*]
[-exclude_lib_cells *collection_of_lib_cells*]
[-objects *macro_list*]
[-bbox { {*llx lly*} {*urx ury*} }]
[-error_view *view_name*]
[-enable_rule_override true | false]

Data Types

list_of_rule_names list
list_of_object_types list
collection_of_lib_cells list
macro_list list
llx float
lly float
urx float
ury float
view_name string

ARGUMENTS

list_of_rule_names

Specifies the list of floorplan rule names to report related violations. By default is checking all floorplan rules set by user.

-object_types ***list_of_object_types***

Report floorplan rule violations on the listed object types. The object types include hard_macro, std_cell_area, soft_macro, block_boundary, core_area. This option cannot be specified with the **-bbox** and **-objects** options.

-lib_cells ***collection_of_lib_cells***

Report floorplan rule violations on the listed lib cells. This option cannot be specified with the **-bbox**, **-objects** and **-exclude_lib_cells** options.

-exclude_rules ***list_of_rule_names***

Specifies the list of floorplan rules that will not report violations.

-exclude_lib_cells ***collection_of_lib_cells***

Specified the lib cells that the violations will not be reported on them. If this option is specified with "-exclude_rules", only the violations match the "-exclude_lib_cells" and "-exclude_rules" will not be reported. This option cannot be specified with the **-bbox**, **-objects** and **-lib_cells** options.

-objects ***macro_list***

Report floorplan rule violations on the listed macros. This option cannot be specified with the **-bbox**, **-object_types**, **-exclude_lib_cells** and **-lib_cells** options.

-bbox { {*llx lly*} {*urx ury*} }

Report floorplan rule violations of macros that fully inside this region. This option cannot be specified with the **-objects**, **-object_types**, **-exclude_lib_cells** and **-lib_cells** options.

-error_view ***view_name***

Specifies the name of the generated error view. If this options is specified, the command saves the error view data to a file in XML format with the specified name. If the view_name file already exists under the current run directory, the command overwrites the file. If a file extension is specified, it must be ".err", otherwise it is ignored. If no extension is specified, the ".err" extension is appended to the file name. If this options is not specified, an attachment is created/modified and saved in to the design database with the file name as "#block_name#_floorplanRules.err" .

-enable_rule_override true | false

When set true, if a lib_cell has its own rule, the corresponding rules for hard_macro would not apply to instances of this lib_cell; In other words, one or more lib_cell rules to an object_type/lib_cell of a certain constraint type overrides any general rule to the same object_type/lib_cell of the same constraint type. Besides this, when it set true, cmd allow lib_cell-routing_blockage enclosure rule to override both hard_macro-routing_blockage enclosure rules and halo rules; Similarly, allow lib_cell-routing_blockage halo rule to override both hard_macro-routing_blockage enclosure rules and halo rules. The default value is false.

DESCRIPTION

This command check floorplan rule violations set by the commands: **set_floorplan_spacing_rules set_floorplan_enclosure_rules set_floorplan_halo_rules set_floorplan_width_rules set_floorplan_area_rules set_track_constraint**

EXAMPLES

The following command checks floorplan rule_1 on hard and soft macros.

```
prompt> check_floorplan_rules rule_1 -object_types {hard_macro soft_macro}
```

Assuming DTCD_lib is the ref lib of a DTCD cell and DTCD cell has the design type of hard macro, the following command reports all macro cell violations except for DTCD cells.

```
prompt> set_floorplan_spacing_rules -name R1 \
  -from_lib_cells DTCD_lib -to_object_types std_cell_area
```

```
prompt> set_floorplan_spacing_rules -name R2 \
  -from_object_types hard_macro -to_object_types std_cell_area

prompt> check_floorplan_rules R2 -enable_rule_override
```

SEE ALSO

set_floorplan_area_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_spacing_rules(2)
set_floorplan_width_rules(2)
set_floorplan_length_rules(2)
remove_floorplan_rules(2)
report_floorplan_rules(2)
set_track_constraint(2)

check_freeze_silicon

Performs feasibility analysis on ECO-cell-to-spare-cell mapping for the current design. This command is typically used in a freeze silicon ECO flow.

SYNTAX

status **check_freeze_silicon**

Data Types

This command has no data types.

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **check_freeze_silicon** command performs feasibility analysis on ECO-cell- to-spare-cell mapping for the current design. The report includes feasibility analysis on ECO cell to spare cell mapping and ECO cell to programmable spare cell (PSC) mapping.

For ECO cells without a PSC type, the report includes:

- The total number of ECO cells and spare cells in the design
- ECO cell mapping feasibility analysis
- Detailed information about the number of ECO cells and spare cells for each library cell

For ECO cells with a PSC type, the report includes:

- The total number of ECO cells and PSC on the design
- ECO cell mapping feasibility analysis for each voltage area and PSC type group
- ECO cells vs. PSC detail report
- Abutted PSC width detail report

When a PSC mapping rule is defined, the "Derived PSC Type", "Derived PSC Width" and "Total PSC Width" columns are shown in the mapping analysis report. The report only includes valid programmable spare cells, while the invalid programmable spare cells due to the PG overlap rule violation are skipped a and the ECO-079 and ECO-090 warning messages are issued.

The command reports feasibility analysis results on ECO cells without a PSC type, then reports ECO cells with a PSC type. The two sections are separated by a section header.

It is recommended that the command is used before the **place_freeze_silicon** command to help you analyze ECO cells and spare cells in your design and to be aware of potential spare cell mapping failures.

The command behavior is the same regardless of the **design.eco_freeze_silicon_mode** application option setting.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows a feasibility check report.

```
prompt> check_freeze_silicon
 ...

================================================================================
Check Freeze Silicon Summary
================================================================================

Type          Eco Number   Spare Number
--------------------------------------------------------------------------------
Total               36          22
--------------------------------------------------------------------------------

================================================================================
ECO Cells Mapping Feasibility
================================================================================

* These references do not have spare cells

Voltage Area      Reference              Eco Number      Spare Number
--------------------------------------------------------------------------------
DEFAULT_VA        BUFFLVTD16                  6             0
DEFAULT_VA        INVLVTD0                    6             0
alu/d2            BUFFLVTD16                  3             0
--------------------------------------------------------------------------------

* These references do not have enough spare cells

Voltage Area      Reference              Eco Number      Spare Number
--------------------------------------------------------------------------------
alu/d2            INVLVTD0                    4             2
cntrl/d1          OAI21HVTD2                  7             2
cntrl/d1          INVLVTD0                    5             4
--------------------------------------------------------------------------------

================================================================================
Eco vs. Spare Cells in detail
================================================================================
```

```
Voltage Area      Reference              Eco Number      Spare Number
-------------------------------------------------------------------------------
DEFAULT_VA        BUFFLVTD16                  6              0
DEFAULT_VA        OAI21HVTD2                  2              4
DEFAULT_VA        INVLVTD0                  6              0
alu/d2            INVLVTD0                4              2
alu/d2            BUFFLVTD16                3              0
alu/d2            OAI21HVTD2                0              7
cntrl/d1          BUFFLVTD16                3              3
cntrl/d1          OAI21HVTD2                7              2
cntrl/d1          INVLVTD0                5              4
-------------------------------------------------------------------------------

****************************************
Report : ECO PSC Feasibility Check
Design : mult36
 ...
****************************************
================================================================================
Check Freeze Silicon Summary
================================================================================

Type          Eco Number    PSC Number
--------------------------------------------------------------------------------
Total              173         2670
--------------------------------------------------------------------------------
* PSC: Programmable Spare Cell

================================================================================
ECO Cells Mapping Feasibility
================================================================================

Voltage Area  PSC Type        Eco Width     PSC Width   Feasibility Check
--------------------------------------------------------------------------------
DEFAULT_VA    1              10.2600       58.5900            FAIL
DEFAULT_VA    2               8.1000      620.2800            PASS
mult_26/FS_1/d1
          1              50.4000        3.9600             FAIL
mult_26/FS_1/d1
          2              81.0000      181.4400             PASS
--------------------------------------------------------------------------------
* Eco Width: total width of eco cells for the voltage area and PSC type group.
* PSC Width: total width of PSC cells for the voltage area and PSC type group.
* Width: in micron

================================================================================
Eco vs. PSC in detail
================================================================================

Voltage Area  PSC Type      Reference
  Eco Number     Eco Width     PSC Number     PSC Width
--------------------------------------------------------------------------------
DEFAULT_VA    1           ND3SKND0P75BWP16P90

  10          4.5000       0           0.0000
```

```
DEFAULT_VA   1          ND3SKND2BWP16P90

  8          5.7600      0          0.0000
DEFAULT_VA   1          FILL3BWP16P90
  0          0.0000      151        40.7700
DEFAULT_VA   1          FILL1BWP16P90
  0          0.0000      198        17.8200
DEFAULT_VA   2          IOAI21D4BWP16P90
  5          8.1000      0          0.0000
DEFAULT_VA   2          FILL4BWP16P90
  0          0.0000      1658       596.8800
DEFAULT_VA   2          FILL2BWP16P90
  0          0.0000      130        23.4000
mult_26/FS_1/d1
         1          ND3SKND0P75BWP16P90
  80         36.0000     0          0.0000
mult_26/FS_1/d1
         1          ND3SKND2BWP16P90
  20         14.4000     0          0.0000
mult_26/FS_1/d1
         1          FILL1BWP16P90
  0          0.0000      11         0.9900
mult_26/FS_1/d1
         1          FILL3BWP16P90
  0          0.0000      11         2.9700
mult_26/FS_1/d1
         2          IOAI21D4BWP16P90
  50         81.0000     0          0.0000
mult_26/FS_1/d1
         2          FILL2BWP16P90
  0          0.0000      14         2.5200
mult_26/FS_1/d1
         2          FILL4BWP16P90
  0          0.0000      497        178.9200
---------------------------------------------------------------------------

===========================================================================
Abutted PSC Width Information
===========================================================================

Voltage Area     PSC Type          Abutted PSC Width          Count
---------------------------------------------------------------------------
DEFAULT_VA       1                    0.0900             198
DEFAULT_VA       1                    0.2700             151
DEFAULT_VA       2                    0.1800              51
DEFAULT_VA       2                    0.3600             205
DEFAULT_VA       2                    0.5400              23
DEFAULT_VA       2                    0.7200             147
DEFAULT_VA       2                    0.9000              31
DEFAULT_VA       2                    1.0800              49
DEFAULT_VA       2                    1.2600              14
DEFAULT_VA       2                    1.4400              41
DEFAULT_VA       2                    1.6200               3
DEFAULT_VA       2                    1.8000              16
DEFAULT_VA       2                    1.9800               2
DEFAULT_VA       2                    2.1600              10
```

```
DEFAULT_VA      2               2.5200          15
DEFAULT_VA      2               2.8800          6
DEFAULT_VA      2               3.0600          1
DEFAULT_VA      2               3.2400          3
DEFAULT_VA      2               3.4200          1
DEFAULT_VA      2               3.6000          2
DEFAULT_VA      2               3.9600          2
DEFAULT_VA      2               4.1400          1
DEFAULT_VA      2               4.6800          2
DEFAULT_VA      2               5.0400          2
DEFAULT_VA      2               5.7600          1
DEFAULT_VA      2               6.8400          2
DEFAULT_VA      2               7.5600          3
DEFAULT_VA      2               7.7400          1
DEFAULT_VA      2               7.9200          1
DEFAULT_VA      2               8.1000          1
DEFAULT_VA      2               8.2800          1
DEFAULT_VA      2               8.8200          1
DEFAULT_VA      2               9.3600          1
mult_26/FS_1/d1 1               0.0900          11
mult_26/FS_1/d1 1               0.2700          11
mult_26/FS_1/d1 2               0.1800          1
mult_26/FS_1/d1 2               0.3600          3
mult_26/FS_1/d1 2               0.5400          4
mult_26/FS_1/d1 2               0.7200          2
mult_26/FS_1/d1 2               0.9000          6
mult_26/FS_1/d1 2               1.0800          2
mult_26/FS_1/d1 2               1.4400          3
mult_26/FS_1/d1 2               1.8000          1
mult_26/FS_1/d1 2               2.1600          1
mult_26/FS_1/d1 2               2.5200          1
mult_26/FS_1/d1 2               3.0600          1
mult_26/FS_1/d1 2               3.2400          1
mult_26/FS_1/d1 2               3.6000          3
mult_26/FS_1/d1 2               3.7800          1
mult_26/FS_1/d1 2               3.9600          2
mult_26/FS_1/d1 2               5.7600          1
mult_26/FS_1/d1 2               6.1200          1
mult_26/FS_1/d1 2               6.8400          2
mult_26/FS_1/d1 2               7.2000          2
mult_26/FS_1/d1 2               8.8200          1
mult_26/FS_1/d1 2               10.4400         1
mult_26/FS_1/d1 2               12.6000         1
mult_26/FS_1/d1 2               12.9600         1
mult_26/FS_1/d1 2               14.0400         1
mult_26/FS_1/d1 2               14.7600         1
mult_26/FS_1/d1 2               15.8400         1
--------------------------------------------------------------------------------

Feasibility check completed.
1
```

The following example shows mapping feasibility analysis report when a PSC mapping rule is defined.

```
================================================================================
ECO Cells Mapping Feasibility
================================================================================
```

```
                                                    Derived
Voltage Area  PSC Type      Eco Width     PSC Width      PSC Type
  Derived        Total
  PSC Width     PSC Width Feasibility Check
--------------------------------------------------------------------------------
DEFAULT_VA   1            36.1760      77.8240        2, 3
   104.5760      182.4000         PASS
DEFAULT_VA   2            21.8880      77.8240        1, 3
   104.5760      182.4000         PASS
DEFAULT_VA   3            12.7680      13.3760        1, 2
   51.6800       65.0560         PASS
--------------------------------------------------------------------------------
* Eco Width: total width of eco cells for the voltage area and PSC type group.
* PSC Width: total width of PSC cells for the voltage area and PSC type group.
* Derived PSC Type: derived PSC types for the voltage area and PSC type group
according to user defined PSC mapping rule.
* Derived PSC Width: total width of derived PSC cells.
* Total PSC Width: sum of PSC width and derived PSC width.
* For details please refer to report_programmable_spare_cell_mapping_rule.
* Width: in micron
```

SEE ALSO

place_freeze_silicon(2)
map_freeze_silicon(2)
report_programmable_spare_cell_mapping_rule(2)

check_hier_design

Checks a hierarchical design for possible issues related to top-level closure. Messages generated by this command are captured in an EMS database file and in the log file.

SYNTAX

status **check_hier_design**
[-stage timing | pre_placement | pre_clock_tree | pre_route | pre_compile]
[-references *reference_design_names*]

Data Types

reference_design_names list

ARGUMENTS

-stage timing | pre_placement | pre_clock_tree | pre_route | pre_compile

Performs atomic checks for the specified design stage.

-stage timing performs timing-related checks.

-stage pre_placement performs both timing and pre-placement related checks.

-stage pre_clock_tree performs timing, pre-placement, and pre-clock-tree checks.

-stage pre_route performs pre-route checks.

-stage pre_compile performs checks to ensure that blocks linked to top are ready for compile.

By default, only timing checks are run if this option is not specified.

-references *reference_design_names*

Specifies the names of design references, for physical hierarchies in the top-level design, on which to run checks. By default, all physical hierarchies are checked if this option isn't specified.

DESCRIPTION

This command performs design checks for use in the top-level closure flow. The command writes the Error, Warning and Info messages to Enhanced Messaging System(EMS) database file and into a text log file.

EXAMPLES

The following example performs timing-related checks.

```
prompt> check_hier_design -stage timing
```

The following example performs pre-placement checks on the RISC_CORE block.

```
prompt> check_hier_design -stage pre_placement -references RISC_CORE
```

The following example performs timing-related checks and writes the results to the EMS database named check_hier.ems

```
prompt> create_ems_database check_hier.ems
prompt> check_hier_design -stage timing
prompt> save_ems_database -all
```

SEE ALSO

check_design(2)
create_ems_database(2)
save_ems_database(2)

check_host_options

Checks that the specified host option is working properly in your environment.

SYNTAX

status **check_host_options**
-host_options *host_option_name*
[-work_dir *directory*]

Data Types

host_option_name string
directory string

ARGUMENTS

-host_options *host_option_name*

Specifies the name of the host option to be checked. Host options are created using the **set_host_options** command. This is a required option.

-work_dir *directory*

Specifies a working directory where various data and log files will be written. By default, the directory is ./work_dir. This directory must be a network disk accessible by all workers.

DESCRIPTION

This command tests the specified distributed processing host option to ensure that the host option is correctly specified for your environment. The command launches a worker process using the host options. This worker process only informs the master process that it is alive and exits.

You should use this command to test your host option settings before beginning a distributed processing run. If this command does not complete successfully, do not use the given host option with any other command because it will not work. Instead, determine the reason for the failure and fix the given host option, or run without distributed processing.

If you have problems getting this command to complete successfully, try first using a simpler host option. You can also try using rsh rather than LSF or GRD.

EXAMPLES

This simple example sets the host option named myLSFtest for LSF distributed processing, then checks the setting.

```
prompt> set_host_options -name myLSFtest -submit_command "bsub"
1
prompt> check_host_options -host_options myLSFtest
Checking host option: myLSFtest
check_host_options successful.
```

SEE ALSO

set_host_options(2)

check_io_placement

Performs I/O placement checking.

SYNTAX

list **check_io_placement**
[-cells *cell_collection*]
[-matching_types *matching_type_collection*]
[-io_guides *io_guide_collection*]
[-pad_assignment_file *file*]
[-filename *filename*]
[-output_directory *directory*]
[-error_view *view_name*]
[-overlap all | pad2pad | pad2filler | filler2filler | bump | corner2pad]
[-bump_assignment]
[-pad_to_guide_assignment]
[-signal_constraints]
[-power_constraints]
[-unplaced]
[-flipping]
[-gap]
[-min_pitch]

Data Types

cell_collection list
matching_type_collection list
io_guide_collection list
file string
filename string
directory string
view_name string

ARGUMENTS

-cells *cell_collection*

Specifies cell instances, such as bumps, pads, fillers, for I/O placement checking. If this option is used, only the specified cells are checked. If this option is used but the specified cells are not valid objects for some specified checks, the specified cells are ignored for those checks. Option **-cells** cannot be used together with option **-signal_constraints**, **-power_constraints**, **-min_pitch**, and **-gap**.

-matching_types *matching_type_collection*

Specifies user-defined matching types for I/O placement checking. If this option is used, only the specified matching types are checked. If this option is used and the specified matching types are not valid objects for the specified checks, this specified matching types are ignored for those checks. Option **-matching_types** cannot be used together with option **-pad_to_guide_assignment**, **-signal_constraints**, **-power_constraints**, **unplaced**, **-min_pitch**, **-gap** and **-flipping**.

-io_guides ***io_guide_collection***

Specifies the I/O guides for I/O placement checking. If this option is used, only the specified I/O guides are checked. If this option is used and the specified guides are not valid objects for the specified checks, this specified matching types are ignored for those checks. Option **-io_guides** cannot be used together with option **-bump_assignment**, **unplaced** and **-flipping**.

-pad_assignment_file ***file***

Specifies the file which contains a list of I/O pad cells and corresponding I/O guides that are allowed for the I/O pad. Entries in this file specify a constraint that limits the pad placement to only the listed I/O guide candidates. If the file is not specified or file cannot be opened, the limitations in file will not be checked. This option can only be used with option **-pad_to_guide_assignment**.

The pad assignment file lists the constraints for the specified I/O pads. Each line contains one or more pad names followed by a list of allowed I/O guide names. I/O guides not in the list cannot be used to place the pads. If more than one pad is listed on one line, the pads must be placed within a pair of curly braces. No curly braces are needed for I/O guides. Each line must end with a semicolon.

```
{pad1 pad2} guide1 guide2;
pad3 guide1 guide3 guide0;
```

-filename ***filename***

Specifies the name of the file where the reports of I/O placement checking are written. Only 10 lines are printed to prompt for each report. In the file, the complete reports are shown. The option cannot be used together with **-output_directory** option.

-output_directory ***directory***

Writes individual report files for each check (overlap, bump assignment, and so on) to the specified directory. The report file names are the name of the check. If the directory does not already exist, the command tries to create the directory. If creation is successful and write permissions are appropriate, the output is written to individual files created in this directory instead of the console. The option cannot be used together with the **-filename** option.

-error_view ***view_name***

Specifies the name of the generated error view. The command saves the error view data to a file in XML format with the specified name. No attachment is created nor saved in to the design database. If the view_name file already exists under the current run directory, the command overwrites the file.

If this option is omitted, the error view data is stored as an attachment to the design database. When the design is closed and saved, the error view data is saved to the design database as well. The saved error view data does not change until the next *check_io_placement* command.

-overlap all | pad2pad | pad2filler | filler2filler | bump | corner2pad

Checks different kinds of overlaps, including pad-pad overlap, pad-filler overlap, filler-filler overlap, bump overlap and corner-pad overlap. One or multiple valid strings must be provided to enable corresponding checks. The valid strings are all, pad2pad, pad2filler, filler2filler, bump. If "all" is specified, the other string is ignored and all kinds of overlap are checked.

If a certain type of overlap checking is specified and invalid objects are specified, only the valid objects are checked and the invalid objects are ignored. For example, if bump overlap checking is specified and some bumps cells and filler cells are specified, the filler cells are ignored.

-bump_assignment

Checks if bump assignment after place_io is consistent with defined matching types (if any) and logical nets. This option works alone or together with **-cells** and **-matching_types**. If the option is specified by itself, all bumps are checked for correct bump

assignment. If the other type of objects are specified, they are ignored for this checking.

-pad_to_guide_assignment

Checks if current pad-to-guide assignment is consistent with the definition. If the option is specified by itself, all pads are checked for placement on the correct guides. Use the **-cells** option to specify the pads to check. If **-io_guides** is specified, all pads that should be placed on specify I/O guides are checked. To use together with **-pad_assignment_file** option, a pad assignment file must be provided. All the pads mentioned in the file are checked. The other type of objects are ignored for this checking.

-signal_constraints

Checks if there are signal I/O pads violating constraints set by **set_signal_io_constraints**. Signal I/O constraints include order, pitch, spacing, offset of pads and boundary-spacing. If the option is specified by itself, all I/O guides with signal I/O constraints are checked.

Note that option **-cells** cannot be used with **-signal_constraints**, as signal I/O constraints can only be set with guides. And in some cases, such as specifying one cell to check for signal pads order, it does not make sense.

-power_constraints

Checks if there are power pads violating constraints set by **set_power_io_constraints**. Power I/O constraints include share, ratio, spacing and offset.

Note that option **-cells** cannot be used with **-power_constraints**, as power I/O constraints can only be set with guides. And in some cases, such as specifying one cell to check for power pads ratio, it does not make sense.

-unplaced

Checks if there are any unplaced pads. If this option is specified by itself, all pads are checked if it is unplaced. If other types of cells are specified, such as fillers, those cells are still checked but with a warning indicating those cells are not pads. If no cells are specified, by default only all pads are checked.

-flipping

Checks if the I/O pads are consistent with the is_flipped attribute as set in the design. If a pad is not placed on any guide, the pad is also considered to violate placement.

-gap

Checks if there is a gap between any two adjacent cells on I/O guides. The unfilled space that is outside the length of the I/O guides are not considered as gap. The valid cell types include pads, corners, fillers and break cells.

-min_pitch

Checks if the min_pitch set with I/O guides are honored by place_io.

DESCRIPTION

This command performs one or multiple kinds of I/O placement checks and return a collection of cells with violations. All options are optional. If no option is used, the command runs all checks. As each checking has different valid objects and options to work with, each check finds the valid objects for the check within the specified objects. Each check ignores the options that specify invalid objects. If the objects specified by an option is not valid for any checking, a warning is issued to indicate the option is not used.

By default, the reports of checks are printed to the console. Each check is written out in a section in report and each section can shows 10 lines of violations at most. You can choose to write out the complete report to files or a directory with options.

EXAMPLES

The following example checks bump assignment for all bumps and signal I/O constraints for specified guides.

```
prompt> check_io_placement -bump_assignment -signal_constraints -io_guides [get_io_guides *]
```

SEE ALSO

create_io_guide(2)
create_matching_type(2)
place_io(2)
set_power_io_constraints(2)
set_signal_io_constraints(2)

check_legality

Checks the legality of the current placement.

SYNTAX

status **check_legality**
[-verbose]
[-cells *list_of_cells*]
[-check_distance *distance*]
[-output_tcl]

Data Types

list_of_cells collection
distance float

ARGUMENTS

-verbose

Prints a detailed report of all possible violations. By default, the command reports only a summary of all violations.

-cells *list_of_cells*

Specifies the cells to check.

When you use this option, the **check_legality** command performs legality check only on the specified cells and cells within the checking distance, which reduces runtime.

By default, the checking distance is three row heights. To adjust the checking distance, use the **-check_distance** option.

-check_distance *distance*

Specifies the checking distance in number row heights. The specified distance must be a float value greater than or equal to zero.

The checking distance determines the neighboring cells to include in legality checking when you use the **-cells** option.

By default, the distance is three row heights.

This option can be used only with the **-cells** option.

-output_tcl

Writes a Tcl file that contains all illegal cells with different violation types. The current process ID (pid) is appended to the file name so the tool can determine if the Tcl file was created in this session.

This option applies only to the classic legalizer. When the advanced legalizer is enabled (the

place.legalize.enable_advanced_legalizer application option is **true**), the command does not write a Tcl file. Instead, the command writes an error file that you can view in the GUI error browser.

By default, no Tcl file is generated.

If you are using the classic legalizer and you use this option, by default, the generated Tcl file contains all moveable cells with violations. You can change this by modifying two variables in the Tcl file, GLOBAL_VIOLATION_TYPE and GLOBAL_VIOLATION_MODE. For information about the valid values for these variables, see the header section at the beginning of the Tcl file.

You can use the generated Tcl file for different purposes, such as

- Fixing violations based on a specific violation type rather than all violation types. For example, use this file to select only overlapped cells and leave other violations as-is.
- Highlighting all violated cells or specific types of violations and avoid parsing the output of the **check_legality -verbose** command.
- Selecting violated cells based on moveable, application fixed, or fixed type. For example, you can select only CTS spacing violations marked as application fixed.
- Using the script information with your own scripts after the collection is created. The generated collection can be used with your own scripts.

DESCRIPTION

The **check_legality** command checks the legality of the current placement and output a report of violation statistics.

By default, the command runs the classic legalizer. To run the advanced legalizer, set the **place.legalize.enable_advanced_legalizer** application option to **true**.

The reported violations can be classified as

- Cells that are not on rows
- Cells overlapping each other
- Cells overlapping blockages
- Cells with orientation that is not allowed for the row on which the cell is placed
- Cells with a core site type not equal to that of the row on which the cell is placed
- Cells with power strap violations
- Cells with advanced technology (20 nm and below) violations
- Cells with placement spacing rule violations

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example checks the legality of all cells and prints a summary of all violations.

prompt> **check_legality**

The following example checks all BUF* cells in the design.

prompt> **check_legality -cells [get_cells BUF*]**

The following example checks cells in the my_cell_list collection.

prompt> **check_legality -cells $my_cell_list**

The following example checks cells added in the my_cell_list collection and cells within 5.5 row heights.

prompt> **check_legality -cells $my_cell_list -check_distance 5.5**

The following example checks all BUF* cells in the design using the default three row height checking distance and writes out a Tcl file including all violated cells.

prompt> **check_legality -cells [get_cells BUF*] -output_tcl**

SEE ALSO

create_placement(2)
legalize_placement(2)
place.legalize.enable_advanced_legalizer(3)

check_legalizer_sanity

Does a series of sanity checks on the design to handle dirty data input.

SYNTAX

status **check_legalizer_sanity**

DESCRIPTION

The **check_legalizer_sanity** command performs a series of sanity checks on the current design (eg. - testing the library, user input design constraints) to avoid legalizer issues.

Running this command before running legalize_placement helps user identify dirty (bad design) inputs that might cause issues like long runtime or hang later, beforehand. The output generated is in the form of Error/Warning messages.

This command helps the user to save time by debugging the cause before legalization runs into problems at later stages; and decide how to proceed.

EXAMPLES

The following example runs the **check_legalizer_sanity** command:

```
prompt> check_legalizer_sanity
```

SEE ALSO

check_legality(2)

check_license

Checks the availability of the specified license feature.

SYNTAX

status **check_license**
feature_name
[-quiet]

Data Types

feature_name string

ARGUMENTS

feature_name

Feature name to be checked for existence in the license server.

-quiet

Do not print any error message if the check failed.

DESCRIPTION

The **check_license** command checks for the existence of the feature in the license server. It does not check out the license feature. If the feature exists on the server, the command returns 1. Otherwise, the command prints an error message and returns 0. The printing of this message can be suppressed via **-quiet** option.

The **list_licenses** command provides a list of the features that are currently checked out.

EXAMPLES

This example checks the availability of the Fusion-Compiler-AG license feature:

```
prompt> check_license Fusion-Compiler-AG
1
```

SEE ALSO

get_licenses(2)
list_licenses(2)
remove_licenses(2)

check_lvs

Checks for inconsistencies in the physical layout of the current design.

SYNTAX

status **check_lvs**
[-nets *collection_of_nets*]
[-checks {short open floating_routes all}]
[-max_errors *maximum_number_of_errors*]
[-check_child_cells true | false]
[-exclude_child_cell_types *exclude_types*
[-check_zero_spacing_blockages true | false]
[-check_top_level_blockages true | false]
[-open_reporting { bounding_box | detailed }]
[-report_floating_pins true | false]
[-treat_terminal_as_voltage_source true | false]

Data Types

collection_of_nets collection
maximum_number_of_errors integer
exclude_types list

ARGUMENTS

-nets *collection_of_nets*

Specifies the nets to check. By default, all nets are checked.

-checks {short open floating_routes all}

Specifies the types of errors to check as follows:

- **short:** check for shorts between nets.
- **open:** check for open nets.
- **floating_routes:** check if any floating routes exist.
- **all:** check all error types above

By default, all error types are checked.

-max_errors *maximum_number_of_errors*

Specifies the maximum number of errors for each type of violations specified with the **-checks** option. Set the value to 0 to specify

an unlimited error count. The default is 20.

-check_child_cells true | false

Specifies whether to check shorts between child cells and top-level shapes. If true, all shapes in child cells are checked. The default is false.

-exclude_child_cell_types {*exclude_types*}

Specifies cell types to ignore when checking shorts between child cell and top-level shapes. Specify one or more of the following cell types: **abstract**, **analog**, **black_box**, **corner**, **cover**, **diode**, **end_cap**, **fill**, **filler**, **flip_chip_driver**, **flip_chip_pad**, **lib_cell**, **macro**, **module**, **pad**, **pad_spacer**, **physical_only**, and **well_tap**.

You must specify the **-check_child_cells** option with this option.

-check_zero_spacing_blockages true | false

Specifies whether to check shorts on zero spacing blockages. The default is false.

-check_top_level_blockages true | false

Specifies whether to check shorts on top-level blockages. The default is true.

-open_reporting { bounding_box | detailed }

Specifies how open violations are shown in the error browser.

- **bounding_box:** Report bounding box of open net
- **detailed:** Report the detailed open locations in the error browser; it might require longer runtime to find detailed open locations

When the number of connected components of an open net is more than 100 and the **-open_reporting** option is set to **detailed**, 100 approximated open locations will be reported. The default is bounding_box.

-report_floating_pins true | false

Specifies whether to report floating pins when checking open or floating_routes violations. The default is false.

-treat_terminal_as_voltage_source true | false

When set to true, the check_lvs command assumes that all terminals of pg nets are voltage sources. The default is false.

DESCRIPTION

The **check_lvs** command detects physical layout versus schematic errors. Violations discovered by the checker are available as an error set for browsing with the GUI error browser.

The command can use multiple CPU cores to generate the report based on the **set_host_options -max_cores** command setting. If this value is not set, the command runs on a single core.

The command generates an error file named "block_name"_lvs.err. To view the errors on GUI, load the error file by using the Error Browser.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports shorts, opens, and floating route violations for all nets in the design.

prompt> **check_lvs**

The following example reports only short violations for all nets in the design.

prompt> **check_lvs -checks short**

The following example reports short, and open violations for net N1, N2, and N3.

prompt> **check_lvs -nets {N1 N2 N3} -checks {short open}**

The following example reports short violations without checking shapes in child cell.

prompt> **check_lvs -checks short -check_child_cells false**

The following example reports open violations in detail on error browser.

prompt> **check_lvs -checks open -open_reporting detailed**

SEE ALSO

check_routes(2)
set_host_options(2)

check_mib_alignment

Checks the alignment of each multiply instantiated block (MIB) instance against power straps, ground straps, standard cell rows, and wire tracks in the current design.

SYNTAX

int **check_mib_alignment**
[-pg]
[-cell_row]
[-wire_tracks]
[-layers *layers*]
[-blocks *blocks*]
[-verbose]

Data Types

layers collection
blocks collection

ARGUMENTS

-pg

Checks the alignment of each MIB instance against the power and ground straps that intersect MIB instances in the current design. If any MIB instances are not aligned correctly with power and ground straps, the command issues a warning message with the location of the misalignment.

-cell_row

Checks the alignment of each MIB instance against standard cell rows and cell row site which intersect the MIB in the current design. If any MIB instances are not aligned correctly with cell rows or cell row site, the command issues a warning message.

-wire_tracks

Checks the alignment of each MIB instance against wire tracks in the current design. If any MIB instances are not aligned correctly with wire tracks, the command issues a warning message.

-layers *layers*

Specifies the layers on which to check the alignment of MIB instances against power straps, ground straps, and wire tracks. This option is used with the **-pg** or **-wire_tracks** options.

-blocks *blocks*

Specifies the reference block name to which the MIB instances belong, for checking the alignment of MIB instances against power straps, ground straps, and wire tracks.

-verbose

Prints complete misalignment report for the checked MIB instances. Without this option, the command prints a compact report (default).

DESCRIPTION

This command checks alignment of each MIB instance against power straps, ground straps, cell rows, and wire tracks respectively or checks all of them without corresponding options. The misaligned MIB instances cannot be pushed down successfully. Misalignment messages are reported in GUI->Window->Message Browser Window when EMS database is created using command create_ems_database.

EXAMPLES

The following example checks for power and ground strap alignment across all the MIB instances in the design.

```
prompt> check_mib_alignment -pg -verbose
```

The following example checks for MIB alignment against wire tracks.

```
prompt> check_mib_alignment -wire_tracks -verbose
```

SEE ALSO

push_down_objects(2)
report_mibs(2)

check_mib_for_pin_placement

Checks multiply instantiated block (MIB) instances for potential pin placement issues.

SYNTAX

string **check_mib_for_pin_placement**
[-swapped_connections]
[-top_level_terminal_locations]
[-asymmetric_connections]
[-cells *list_of_blocks*]
[-nets *net_collection*]

Data Types

list_of_blocks list
net_collection collection

ARGUMENTS

-swapped_connections

Checks and reports pins on MIB instances that do not connect to the corresponding pins across the instances of the reference block. The check can be run before and after pin placement. The check is limited to comparing two-pin nets that connect MIB instances.

-top_level_terminal_locations

Checks and reports top-level terminals that connect to the same MIB pin, but at difference locations. The check can be run before and after pin-placement.

-asymmetric_connections

Checks for pins across MIB instances where the MIB block pin does not connect to the same number or type of pins (asymmetric pins). The check can be run before and after pin placement. Unlike the **-swapped_connections** check, the **-asymmetric_connections** check is not limited to two-pin nets that connect two unique MIB instances. Instead, this check examines all MIB pins in the design for asymmetry. By definition, swapped connections are a subset of asymmetric connections. The command skips pins that are connected to tie-high or tie-low nets. When this option is used the command returns asymmetric connections/pins as well as pins connected to these asymmetric connections to help with further analysis.

-cells *list_of_blocks*

Specifies a list of blocks or cells for checking. The check considers all the MIB blocks in the design for checking, but limits reporting and the collection of pins to the set of blocks specified in the *list_of_blocks*.

-nets *net_collection*

Specifies a collection of nets for mib checking. The check considers all the MIB blocks in the design for checking, but limits reporting and the returned collection of pins to those pins that connect to the specified set of nets. If this option is not specified the command looks at all MIB pins.

DESCRIPTION

This command checks and reports potential multiply instantiated block (MIB) pin placement issues. If an issue is detected, the command returns a collection of pins with potential problems. The command can be used before or after pin placement to detect potential issues with MIB pins. You must specify at least one of **-top_level_terminal_locations** or **-swapped_connections** options.

The **set_editability** can enable/disable this command for blocks or levels of physical hierarchy. This command has no effect on the blocks that are disabled by **set_editability**.

MIB pins that are dangling or tied-off are ignored by this command. That is the comparisons of MIB pin connectivity are limited to those pins that are neither dangling nor tied off.

EXAMPLES

The following example checks for swapped MIB pin connections pins on all blocks.

```
prompt> check_mib_for_pin_placement -swapped_connections
```

SEE ALSO

place_pins(2)
report_feedthroughs(2)
set_editability(2)

check_multibit_library

Reports the compatible multibit and single bit cells in the library for banking and debanking.

SYNTAX

status **check_multibit_library**
[-banking]
[-debanking]
[-generate_map_file *file_name*]

ARGUMENTS

-banking

Reports all compatible multibit lib cells for each single bit lib cell for banking.

-debanking

Reports all compatible single bit lib cells for each multibit lib cells for debanking.

If neither \fb-banking nor **-debanking** option is specified, the \fb-banking will be specified by default.

-generate_map_file *file_name*

Specifies the name of output file that contains the mapping information. The file contains a series of function equivalent lib cell groups for single bit cells. The file also contains the mapping of multibit bit-width to the combination of lower multibit lib cells in each function equivalent group.

DESCRIPTION

This command lists all the compatible lib cells for banking and debanking. The report also includes the reasons of incompatibility. The report lists the compatible and incompatible multibit lib cells for each single bit lib cell. On the other hand, the report lists the compatible and incompatible single bit lib cells for each multibit lib cell if -debanking option is specified.

If the -generate_map_file is specified, the output file will be generated. The output file contains a collection of the function equivalent lib cells. The output file also shows how to use the lower multibit lib cells to compose a multibit cell whose bit-width is under 32 for each function equivalent group.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example generate a compatible lib cells report for banking and a output mapping file.

```
prompt> check_multibit_library -generate map_file.txt

****************************************
Report : check_multibit_library
Flow   : BANKING
****************************************


--------------------------------------------------------------------------------
Single bit Lib cell          Compatibility      Multi bit Lib cell
--------------------------------------------------------------------------------
SB_Reg_Lib1                  COMPATIBLE         MB_Reg_Lib1
                                                MB_Reg_Lib2
                                                MB_Reg_Lib3


SB_Reg_Lib2                  VOLT INCOMPATIBLE  MB_Reg_Lib4
                                                MB_Reg_Lib5
                                                MB_Reg_Lib6


Singlebit with NO Multibit Equivalents
--------------------------------------------------------------------------------
SB_Reg_Lib3
SB_Reg_Lib4
SB_Reg_Lib5


The map_file.txt contains the information as below:
FUNC_GROUP_1 { SB_Reg_Lib6 SB_Reg_Lib7 SB_Reg_Lib8 }
 8 { 1 MB_Reg_Lib7 }
 4 { 1 MB_Reg_Lib8 }
 2 { 1 MB_Reg_Lib9 }
```

The FUNC_GROUP_1 { SB_Reg_Lib6 SB_Reg_Lib7 SB_Reg_Lib8 } shows the function equivalent group for single bit lib cells.

2 { 1 MB_Reg_Lib9 } shows the target multibit bit-width 2 would be composed by 1
of MB_Reg_Lib9 multibit lib cells.

SEE ALSO

identify_multibit(2)
create_multbiit(2)
split_multibit(2)

check_mv_design

Checks for violations in a multi-voltage design.

SYNTAX

status **check_mv_design**
[-max_message_count *message_count*]
[-erc_mode]
[-power_connectivity]
[-voltage_threshold *threshold*]
[-interface_only]

Data Types

message_count integer
threshold float

ARGUMENTS

-max_message_count *message_count*

Set the maximum number of occurrences for each error type. If you set it to zero or 'all', that means the number of occurrences of the message is unlimited. If this option is not specified, the limit is 20 by default.

-erc_mode

If specified, the command will run in electrical-rule-checking mode and ignore any UPF strategies specified in the power intent. If this option is not specified, the command will consider UPF strategies.

-power_connectivity

If specified, the command will check pg nets and power and ground connections of the design.

-voltage_threshold *threshold*

Set the voltage threshold for level shifting violations. If this option is not specified, the threshold is zero by default.

-interface_only

If specified, the command will check violations related to block boundary interface of current block and sub-block(s) only and other internal violations will be excluded.

DESCRIPTION

The **check_mv_design** command checks multi-voltage related electrical rules for the design, and issues error and warning messages as appropriate.

EXAMPLES

The following command prints all multi-voltage related violations in the design.

```
prompt> check_mv_design -power_connectivity -max_message_count all

--------- Power domain rule ----------
No errors or warnings.

--------- Supply net rule ----------
Warning: Unable to derive power and ground type for supply net vddx. (UPF-049)
Information: Total 1 UPF-049 violations. (MV-080)
Error: Supply net 'vddx' has no valid driver(s). (MV-011)
Information: Total 1 MV-011 violations. (MV-080)

--------- Supply port rule ----------
Warning: HighConn of supply port 'U11/vdd089' is unconnected. (MV-022)
Warning: LowConn of supply port 'U11/vdd089' is unconnected. (MV-022)
Information: Total 2 MV-022 violations. (MV-080)

--------- Isolation cell rule ----------
No errors or warnings.

--------- Level shifter cell rule ----------
No errors or warnings.

--------- Retention cell rule ----------
No errors or warnings.

--------- Switch cell rule ----------
No errors or warnings.

--------- Isolation rule ----------
Error: The related supply net 'vdd' of driver pin 'in1' is less always-on than
the related supply net 'vdd2' of load pin 'ff2/D'. (MV-013)
Error: The related supply net 'vdd' of driver pin 'clk' is less always-on than
the related supply net 'vdd2' of load pin 'ff2/CK'. (MV-013)
Information: Total 2 MV-013 violations. (MV-080)

--------- Voltage shifting rule ----------
Error: Driver pin 'in1' (related supply net: vdd [0.6V]) cannot drive load pin
'ff2/D' (related supply net: vdd2 [0.5V]) due to voltage difference. (MV-012)
Error: Driver pin 'clk' (related supply net: vdd [0.6V]) cannot drive load pin
'ff2/CK' (related supply net: vdd2 [0.5V]) due to voltage difference. (MV-012)
```

```
Information: Total 2 MV-012 violations. (MV-080)

--------- PG net rule ----------
No errors or warnings.

--------- Tie-off connection rule ----------
Warning: The tie-off connection 'm1' has not been implemented. (MV-027)
Information: Total 1 MV-027 violations. (MV-080)

--------- PG pin rule ----------
No errors or warnings.

--------- Power connectivity rule ----------
No errors or warnings.

--------- Signal pin rule ----------
No errors or warnings.
1
```

SEE ALSO

load_upf(2)

check_netlist

Checks the netlist for any connectivity errors or violations and issues warning messages accordingly.

SYNTAX

int **check_netlist**
[-cells]
[-nets]
[-ports]
[-hierarchical]
[-multiple_designs]
[-summary]

ARGUMENTS

-cells

Specifies that netlist checks to be performed on cell instances. This option involves checking the input and output pin connections of the cell instances as well as the inside connections for hierarchical cell instances.

-nets

Specifies that netlist checks to be performed on nets. This option involves checking for drivers, loads and multiple drivers of a net. It also checks if a net is constant driven or it is three state net.

-ports

Specifies that netlist checks to be performed on ports. This option involves checking of the port connections such as whether an output port is being driven, an input port has a load.

-hierarchical

Performs the checks on the objects across all levels of physical hierarchy. This option is FALSE by default.

-multiple_designs

Reports warning messages related to multiply-instantiated designs.

-summary

Displays a summary of the warning messages instead of one message per warning. This does not affect the way error messages are issued.

DESCRIPTION

The command checks the design for netlist connectivity errors and any design violations. This command gives a preliminary information about the state of the netlist and if there are any potential problems with the netlist. This informs about the potential errors that might occur during the later stages.

The command performs checks cells/nets/ports based on the options provided. The command by default performs checks on cells, net and ports. When no option is given, the **check_netlist** command performs checks for cells, nets and ports of the top level hierarchy. If **-hierarchy** is given, the command would perform the checks across all levels of physical hierarchy. The option **-multiple_designs** flags multiple instances of a module. If a module is instantiated in two different designs, a warning message is issued.

If an open EMS database exists before this command is executed, the messages are saved to the EMS databse. The Message Browser window in GUI provides a summary of the number of messages for each check type as well as each individual message.

This command has no dependency on scenario-specific information.

EXAMPLES

The check_netlist command displays the following information when no option is specified.

```
prompt> check_netlist
****************************************
Report : check_netlist
        -cells
        -nets
        -ports
Design : gecko
 ...

****************************************
Warning: In design 'sub_92', input pin 'sub_92/CI' of hierarchical cell
  'sub_92' has no internal loads and is not connected to any nets. (DCHK-004)
Warning: In design 'sub_92', a pin on submodule 'sub_92' is connected to
  logic 1 or logic 0. (DCHK-005)
Warning: In design 'add_86', input pin 'add_86/CI' of hierarchical cell
  'add_86' has no internal loads and is not connected to any nets. (DCHK-004)
Warning: In design 'add_86', a pin on submodule 'add_86' is connected to
  logic 1 or logic 0. (DCHK-005)
Warning: In design 'add_79', input pin 'add_79/CI' of hierarchical cell
  'add_79' has no internal loads and is not connected to any nets. (DCHK-004)
Warning: In design 'add_79', a pin on submodule 'add_79' is connected to
  logic 1 or logic 0. (DCHK-005)
Warning: In design 'gecko', cell 'EFF1' does not drive any nets. (DCHK-008)
Warning: In design 'sub_92', input port 'sub_92/CI' is unloaded. (DCHK-017)
Warning: In design 'sub_92', net connected to output port 'sub_92/CO' is not
  driven. (DCHK-021)
Warning: In design 'add_86', input port 'add_86/CI' is unloaded. (DCHK-017)
Warning: In design 'add_86', net connected to output port 'add_86/CO' is not
  driven. (DCHK-021)
Warning: In design 'add_79', input port 'add_79/CI' is unloaded. (DCHK-017)
Warning: In design 'add_79', net connected to output port 'add_79/CO' is not
  driven. (DCHK-021)
```

```
Warning: In design 'sub_69', input port 'sub_69/CI' is unloaded. (DCHK-017)
Warning: In design 'sub_69', net connected to output port 'sub_69/CO' is not
  driven. (DCHK-021)
Warning: In design 'add_59', input port 'add_59/CI' is unloaded. (DCHK-017)
Warning: In design 'add_59', net connected to output port 'add_59/CO' is not
  driven. (DCHK-021)
Warning: In design 'sub5/add_357', input pin 'sub5/add_357/CI' of hierarchical
  cell 'sub5/add_357' has no internal loads and is not connected to any nets. (DCHK-004)
Warning: In design 'sub5/add_357', a pin on submodule 'sub5/add_357' is
  connected to logic 1 or logic 0. (DCHK-005)
Warning: In design 'sub5/sub_348', input pin 'sub5/sub_348/CI' of hierarchical
  cell 'sub5/sub_348' has no internal loads and is not connected to any nets. (DCHK-004)
Warning: In design 'sub5/sub_348', a pin on submodule 'sub5/sub_348' is
  connected to logic 1 or logic 0. (DCHK-005)
Warning: In design 'sub5/add_339', input pin 'sub5/add_339/CI' of hierarchical
  cell 'sub5/add_339' has no internal loads and is not connected to any nets. (DCHK-004)
Warning: In design 'sub5/add_339', a pin on submodule 'sub5/add_339'
  is connected to logic 1 or logic 0. (DCHK-005)
Warning: In design 'sub5/add_357', input port 'sub5/add_357/CI' is
  unloaded. (DCHK-017)
Warning: In design 'sub5/add_357', net connected to output port
  'sub5/add_357/CO' is not driven. (DCHK-021)
Warning: In design 'sub3/add_252', input pin 'sub3/add_252/CI' of hierarchical
  cell 'sub3/add_252' has no internal loads and is not connected to any nets. (DCHK-004)
Warning: In design 'sub3/add_252', a pin on submodule 'sub3/add_252' is
  connected to logic 1 or logic 0. (DCHK-005)
Warning: In design 'sub3/add_246', input pin 'sub3/add_246/CI' of hierarchical
  cell 'sub3/add_246' has no internal loads and is not connected to any nets. (DCHK-004)
Warning: In design 'sub3/add_246', a pin on submodule 'sub3/add_246' is
  connected to logic 1 or logic 0. (DCHK-005)
Warning: In design 'sub3/add_252', input port 'sub3/add_252/CI' is
  unloaded. (DCHK-017)
Warning: In design 'sub3/add_252', net connected to output port
  'sub3/add_252/CO' is not driven. (DCHK-021)
Warning: In design 'sub3/add_246', input port 'sub3/add_246/CI' is
  unloaded. (DCHK-017)
Warning: In design 'sub3/add_246', net connected to output port
  'sub3/add_246/CO' is not driven. (DCHK-021)
Information: In design 'gecko', 'flat' net 'out[0]' has multiple drivers. (DCHK-011)
Information: In design 'gecko', 'flat' net 'out[1]' has multiple drivers. (DCHK-011)
Information: In design 'gecko', 'flat' net 'out[8]' has multiple drivers. (DCHK-011)
Warning: In design 'sub5', 'hierarchical' net 'sub5/*Logic0*' has no drivers. (DCHK-010)
Warning: In design 'sub5/add_339', 'hierarchical' net 'sub5/add_339/CI' driven
  by pin 'sub5/add_339/CI' has no loads. (DCHK-009)
Warning: In design 'sub5/sub_348', 'hierarchical' net 'sub5/sub_348/CI' driven
  by pin 'sub5/sub_348/CI' has no loads. (DCHK-009)
Warning: In design 'sub5/add_357', 'hierarchical' net 'sub5/add_357/CI' driven
  by pin 'sub5/add_357/CI' has no loads. (DCHK-009)
Warning: In design 'sub5/add_357', 'hierarchical' net 'sub5/add_357/CO' has no
  drivers. (DCHK-010)
Warning: In design 'sub3', 'hierarchical' net 'sub3/*Logic0*' has no drivers. (DCHK-010)
Warning: In design 'sub3/add_246', 'hierarchical' net 'sub3/add_246/CI' driven
  by pin 'sub3/add_246/CI' has no loads. (DCHK-009)
Warning: In design 'sub3/add_252', 'hierarchical' net 'sub3/add_252/CI' driven
  by pin 'sub3/add_252/CI' has no loads. (DCHK-009)
Warning: In design 'sub3/add_252', 'hierarchical' net 'sub3/add_252/CO' has no drivers. (DCHK-010)
```

```
Warning: In design 'sub1/add_185', 'hierarchical' net 'sub1/add_185/CO' has no drivers. (DCHK-010)
1
```

check_objects_for_push_down

Checks a collection of nets or charging stations for potential problems before running the **push_down_objects** command.

SYNTAX

collection **check_objects_for_push_down**
object_collection
[-cells *cells*]

Data Types

object_collection collection
cells collection

ARGUMENTS

object_collection

Specifies the collection of signal nets or charging stations to be checked. The collection can only include signal nets or charging stations, but not both. This option is required.

Only **signal_routing** objects and charging stations are checked by this command. The **signal_routing** objects include wires, paths, vias, via arrays and terminals that connect to signal nets. The charging stations are non-primary voltage areas that contain repeater buffers.

-cells *cells*

Specifies the block instances to check for pushing down objects. By default, all blocks are checked for push-down readiness.

DESCRIPTION

This command checks signal nets or charging stations for potential issues before pushing them into blocks with the **push_down_objects** command. The check returns a collection of nets or charging stations that might have problems. You can check only one object type, net or charging station, at a time.

If the command finds no problems with any of the objects passed (nets or charging stations), the command returns an empty collection. If the command encounters some problem unrelated to specific checks for objects, the command returns a "0" status rather than a collection.

When nets or shapes of nets are included in the object collection, this command traverses the routing and looks for 1) collinear routes and vias, 2) problems with physical routing that are incompatible with the logical definition of the net, 3) incomplete routing at the top level.

If charging stations are included, the command checks that the voltage area is non-primary and finds the deepest block that the voltage area overlaps. The depth of hierarchy that is examined is controlled by the **set_editability** command. Use this command to define the range of hierarchy depth or the blocks to check. The command checks if that block has the necessary UPF objects (PG supply nets) for receiving that voltage area.

This command uses the option settings specified by the **set_push_down_object_options** command. Set all options for the net or charging station objects that are planned for push-down before running the **check_objects_for_push_down** command.

For net routing, the command checks the routing and determines whether feedthrough ports are required. By default, feedthrough requirements (such as top-level continuity, and logical-vs-physical consistency of the routing) are not checked unless the command detects pin constraints for the net that indicate that the **push_down_objects** command would be free to allow feedthroughs on the net. If so, and the routing requires a feedthrough port on the block, feedthrough requirements are checked for the net.

Feedthroughs for nets can be enabled or disabled by the **set_individual_pin_constraints -nets netname -allow_feedthroughs true**|**false** command.

EXAMPLES

The following example checks the routing objects for nets NETA and NETB for potential push down problems.

```
prompt> check_objects_for_push_down [get_nets {NETA NETB}]
```

The following example checks the charging station named "gas_station" for potential push down problems.

```
prompt> check_objects_for_push_down [get_voltage_areas gas_station]
```

SEE ALSO

push_down_objects(2)
remove_push_down_object_options(2)
report_push_down_object_options(2)
set_push_down_object_options(2)
set_individual_pin_constraints(2)

check_pg_connectivity

Verifies physical connectivity of the power ground network.

SYNTAX

status **check_pg_connectivity**
[-nets *pg_nets*]
[-write_connectivity_file *file_name*]
[-check_macro_pins one | all | none]
[-check_block_pins one | all | none]
[-check_pad_pins one | all | none]
[-check_std_cell_pins one | all | none]
[-max_floating_cluster_size *size*]
[-error_view_name *file_name*]

Data Types

pg_nets collection
file_name string
size integer

ARGUMENTS

-nets *pg_nets*

Specifies the power ground nets to be checked. By default, the command checks all power ground nets.

-write_connectivity_file *file_name*

Specifies the output file name. If specified, the tool writes detailed net connectivity information to the file.

-check_macro_pins one | all | none

Specifies how macro pins are checked for the specified nets. "one" means that a macro is connected if at least one pin shape of the net is connected. "all" means that a macro is connected only if all pin shapes of the net are connected. "none" means that macro pins are not checked. By default, the value is "one".

-check_block_pins one | all | none

Specifies how block pins are checked for the specified nets. "one" means that a block is connected if at least one pin shape of the net is connected. "all" means that a block is connected only if all pin shapes of the net are connected. "none" means that block pins are not checked. By default, the value is "all".

-check_pad_pins one | all | none

Specifies how pad pins are checked for the specified nets. "one" means that a pad is connected if at least one pin shape of the

net is connected. "all" means that a pad is connected only if all pin shapes of the net are connected. "none" means that pad pins are not checked. By default, the value is "one".

-check_std_cell_pins one | **all** | **none**

Specifies how std cell pins are checked for the specified nets. "one" means that a std cell is connected if at least one pin shape of the net is connected. "all" means that a std cell is connected only if all pin shapes of the net are connected. "none" means that std cell pins are not checked. By default, the value is "one".

-max_floating_cluster_size ***size***

Specify the maximum size of floating cluster to be returned by the command and reported by the error browser.

-error_view_name ***file_name***

Specifies the error view name. If not specified, the default error view name will be "[block_name]_floatingPG.err".

DESCRIPTION

This command checks the physical connectivity of the power ground network. The command generates information for floating wires, vias, pads, macro pins, and standard cell pins. The information can be viewed by using the error browser. More detailed info can also be written to a file. The command also returns a collection of floating vias and wires. You can easily remove these floating objects using **remove_objects** command.

EXAMPLES

The following example checks the connectivity of the VSS net, and checks whether all macro power ground pins are connected to the network.

prompt> **check_pg_connectivity -nets VSS -check_macro_pins all**

SEE ALSO

check_pg_missing_vias(2)
compile_pg(2)
connect_pg_net(2)

check_pg_drc

Verifies and reports technology routing design rules violations and illegal overlaps of PG net objects.

SYNTAX

status **check_pg_drc**
[-nets {*collection_of_nets*}]
[-ignore_std_cells]
[-do_not_check_shapes_in_lib_cells]
[-do_not_check_shapes_in_hier_blocks]
[-ignore_clock_nets true | false]
[-check_metal_on_track true | false]
[-ignore_keepout_margins]
[-check_min_metal_area_on_pins true | false]
[-load_routing_of_all_nets]
[-check_detail_route_shapes]
[-coordinates {{*llx lly*} {*urx ury*}}]
[-bottom_layer *min_layer_name*]
[-top_layer *max_layer_name*]
[-no_gui]
[-output *text_file*]

Data Types

collection_of_nets collection or list
llx float
lly float
urx float
ury float
min_layer_name string
max_layer_name string
text_file string

ARGUMENTS

-nets {*collection_of_nets*}

Specifies the nets to verify. By default, all PG nets are checked.

-ignore_std_cells

Prevents the command from loading content of standard library cells. This may be useful, for example, if placement of the cells is not "legalized". By default, the command loads and checks all internal objects of such cells.

-do_not_check_shapes_in_lib_cells

When this option is specified, the command does not report drc errors, which involve internal shapes of same library cell only. Cells under this option include cells of the following design_type: lib_cell, filler, end_cap, well_tap, diode. By default, such errors are reported. For example, small unconnected pin, included into PG net may cause "minimal metal area" error in a report.

-do_not_check_shapes_in_hier_blocks

When this option is specified, the command does not report drc errors, which involve internal shapes of any cell other than library cells described in option "-do_not_check_shapes_in_lib_cells". For example, design_type macro, pad, etc. By default, such errors are reported.

-ignore_clock_nets true | false

Specifies whether routing objects of clock nets should be ignored. By default, the tool ignores clock net objects.

-check_metal_on_track true | false

Specifies whether to check metal-shape alignment with a track on a metal layer (if corresponding rules are set in the technology file). By default, the tool does not do such checking.

-ignore_keepout_margins

When this option is specified, the command does not check any overlaps of routing objects with keepout-margin areas. By default, all such cases are reported as illegal-overlap errors.

-check_min_metal_area_on_pins true | false

Specifies whether the command should check metal area of unconnected pins. By default, insufficient area of such pins is reported as an error.

-load_routing_of_all_nets

This is request to consider existing routing shapes and vias of all nets. By default, routing shapes and vias of non-PG nets are completely ignored.

-check_detail_route_shapes

When this option is specified, the command takes into account routing shapes and vias of PG nets which have usage attribute "detail_route". By default, such shapes are ignored.

-coordinates {{*llx lly*} {*urx ury*} ...}

Specifies a rectangular area where to perform design rule checking. *llx* and *lly* are coordinates that define the lower-left corner of the area. *urx* and *ury* are coordinates that define the upper-right corner. By default, PG DRC checking is performed on the entire design.

When this option is specified, the tool considers objects within the specified rectangle, and nearby objects associated with the PG nets.

-bottom_layer *min_layer_name*

Specifies the bottom routing layer on which to perform DRC checking. If this option is specified, the tool ignores objects on layers lower than *min_layer_name*. By default, the tool checks down to the lowest routing layer of design.

-top_layer *max_layer_name*

Specifies the top routing layer on which to perform DRC checking. If this option is specified, the tool ignores objects on layers higher than *max_layer_name*. By default, the tool checks up to the highest routing layer of design.

-no_gui

Prevents the tool from writing to the GUI error set. If this option is specified, a text report is written to the file specified by the **-output** option. If **-output** is not specified, the tool writes the report to a file named "./pg_routing_errors.txt".

-output ***text_file***

Specifies the name of the output report file. By default, the tool does not write a text report, unless the **-no_gui** or this option is specified. If **-no_gui** is specified and **-output** is omitted, the tool writes the report to a file named "./pg_routing_errors.txt".

DESCRIPTION

This command checks and reports violations of technology design rules and illegal overlaps of objects (shapes, vias and pins) of power and ground nets.

Checking is performed either on entire design area, or in the area specified by the **-coordinates** option.

The checking result from this command is available as an error set for browsing with the GUI error browser, or as text report that is written to a file, or both.

For the text report, the command writes the error type, violated rule, layer, and other details regarding the error as follows.

```
=================
Error type:     insufficient space between two diff-net via-cuts
Violated rule:  Cut-Min_Spacing-Rule
Layer:          CO
Minimal spacing: 0.05
Actual  spacing: 0.0238(southwest - northeast)
Spacing box:    {900.621 412.336} {900.633 412.357}
Via #1:         box = {900.579 412.294} {900.621 412.336}
                net = VDD
Via #2:         box = {900.633 412.357} {900.675 412.399}
            net = VSS
```

The command can use multiple CPU cores to generate the report, based on the *set_host_options -max_cores* command setting. If this value is not set, the command runs on a single core.

EXAMPLES

The following example checks technology design rules for all PG nets in the entire design. The report is generated as an error set for the GUI error browser. The command can use up to 6 CPU cores:

```
prompt> set_host_options -max_cores 6
prompt> check_pg_drc
```

The following example checks only objects of the VDD net on layer MET4. The text format report is written to a file named "./net_vdd_on_M4.drc":

```
prompt> check_pg_drc -no_gui -output ./net_vdd_on_M4.drc -nets VDD \
  -bottom_layer MET4 -top_layer MET4
```

The following example checks rules within the specified rectangular area and generates both kinds of report:

```
prompt> check_pg_drc -coordinates {{287.0 362.0} {412.0 530.0}} \
```

-output ./PG-errors_in_window.txt

SEE ALSO

set_host_options(2)
gui_open_error_data(2)

check_pg_missing_vias

Checks missing vias within the power and ground network.

SYNTAX

status **check_pg_missing_vias**
[-nets *pg_nets*]
[-via_rule_file *rule_file_name*]
[-output_file *output_file_name*]
[-write_default_via_rule_file]
[-shape_use *shape_use_list*]
[-ignore_shield_route]
[-ignore_small_intersections]
[-honor_routing_blockage]

Data Types

pg_nets collection
rule_file_name string
output_file_name string
shape_use_list list

ARGUMENTS

-nets *pg_nets*

Specifies the PG nets to be checked. By default, the command checks all PG nets.

-via_rule_file *rule_file_name*

Specifies the via rule file which includes all the via rules to be checked. By default, the command checks vias between adjacent orthogonal layers.

The via rule file can include multiple via rules.

The supported rules are

- **stack_via**: Checks for stacked vias.

  By default (**false**), this rule is not checked.

- **parallel**: Checks for vias between parallel layers.

  When you specify this rule, you must also specify the pitch with the **pitch** attribute.

  By default (**false**), this rule is not checked.

- **pitch** Specifies the via pitch for parallel wires.
- **macro_pin**: Checks for vias connecting to macro or power-switch PG pins.

  By default (**false**), this rule is not checked.
- **terminal**: Checks for vias from terminal in upper layer to wires in lower layer

  By default (**false**), this rule is not checked.

The via rule format is as follows:

```
viaRuleName {
  bottom_layer: bottom_metal_layer_name (required)
  top_layer: top_metal_layer_name      (required)
  stack_via: true | false
  parallel: true|false
  pitch: number
  macro_pin: true | false
  terminal: true | false
}
```

The following example checks vias between M2 and M7, including stacked vias and macro pins.

```
viaRule1 {
  bottom_layer: M2
  top_layer:    M7
  stack_via:    true
  macro_pin:    true
!}
```

-output_file *output_file_name*

Specifies the output file name. If specified, the tool writes detailed information about missing vias to the file.

-write_default_via_rule_file

Writes the default via rule to defaultViaRule.txt.

-shape_use

If specified, the command will only check vias against shapes with specified shape_use in the option. If not specified, the command will check vias against shapes with all shape_use.

-ignore_shield_route

If specified, the command will not check vias against shapes with shape use shield_route.

-ignore_small_intersections

If specified, the command will not report error if the via site is smaller than the minimum contact code on the layer.

-honor_routing_blockage

If specified, the command will not report error if the via site is covered by routing blockages.

DESCRIPTION

This command checks for missing vias between overlapping regions of different metal layers. The error info can be viewed from error browser or written to a file. You can define via rules to be checked by writing the rules to a file and importing the file with the **-via_rule_file** option.

EXAMPLES

The following example checks the vias for the VSS net based on rules defined in viaRule.txt.

```
prompt> check_pg_missing_vias -nets VSS -via_rule_file viaRule.txt
Check net VSS vias...
Number of missing vias: 0
Checking net VSS vias took 0 seconds.
```

SEE ALSO

create_pg_vias(2)
compile_pg(2)

check_physical_constraints

Checks the design for any issues in the floorplan objects and constraints such as macro or fixed cell instances, site rows, blockages, core/die area, layers, tracks, bounds, and so on.

SYNTAX

int **check_physical_constraints**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The command checks the design for any issues with the physical constraints. Use this command to discover potential issues which could lead to problems later in the flow. Depending on the severity, the command issues an information, warning, or error message.

The command checks whether floorplan data is available or not, and checks for the presence of core and die areas. It also performs various checks on site rows, move bounds, group bounds, layers, tracks, fixed and macro cell instances, and placement blockages.

Basic Floorplan:
The command checks that basic floorplan information, like boundary and core, is defined for the block or that the information is valid.
Checks if the unit tile is missing in reference library.

Site Rows:
Checks if the site height is multiple of the normal site height.
Checks if the site row overlaps another site row.
Checks if the site row needs to be aligned.
Checks if there are both horizontal and vertical site rows.
Checks if it is a non-zero extra spacing in rows which is not supported.

Cell Instance:
Checks if the keepout margin is outside the core area.
Checks if the I/O pad cell instance is in the core area or overlaps with other cell instance.
Checks if the orientation of cell instance is legal.
Checks if the macro is placed and fixed.
Checks if the macro is out of the core area.
Checks if the macro is overlapping with other macro or blockage.

Placement Blockages/Bounds:
Checks if the movebound is completely covered by blockage.

Checks if the placement blockage is partially or completely outside the core area.
Checks if the move bound is partially or completely overlap with other.
Checks if the utilization of group bound/move bound is above 90%.

Layers/Tracks:
Checks if the tracks are defined.
Checks if the track crosses core area.
Checks if the preferred routing direction is defined.
Checks if the consecutive metal layers have the same preferred routing direction.
Checks if the layer spacing is greater than the pitch minus width.
Checks if a layer has non-PG shapes/shape patterns on it.
Checks if a layer is defined in the technology file.

If an open EMS database exists before this command is executed, the messages are saved to the EMS database. The Message Browser window in the GUI provides a summary of the number of messages for each check type as well as each individual message.

This command has no dependency on scenario-specific information.

EXAMPLES

The **check_physical_constraints** command displays the following information.

```
prompt> check_physical_constraints
****************************************
Report : check_physical_constraints
Design : mult36
Version: L-2016.03-SP3-BETA
Date   : Fri Jun  3 03:10:22 2016
****************************************
Warning: The spacing of layer 'VTL_N' is greater than the difference
  of the pitch and width of the layer. (DCHK-105)
Warning: The spacing of layer 'VTL_P' is greater than the difference
  of the pitch and width of the layer. (DCHK-105)
Warning: The spacing of layer 'PO' is greater than the difference of
  the pitch and width of the layer. (DCHK-105)
Information: The layer 'M8' does not contain any PG shapes. (DCHK-104)
Information: The layer 'M9' does not contain any PG shapes. (DCHK-104)
Warning: The orientation of cell instance 'tapfiller_TAPCELLBWP16P90_0'
  does not match any legal orientations. (DCHK-103)
Warning: The orientation of cell instance 'tapfiller_TAPCELLBWP16P90_1'
  does not match any legal orientations. (DCHK-103)
1
```

check_pin_placement

Performs pin placement checking.

SYNTAX

string **check_pin_placement**
[-alignment true | false]
[-alignment_tolerance_distance *tolerance_distance*]
[-layer_mismatch_only true | false]
[-blocked_only true | false]
[-alignment_report_file *file*]
[-alignment_histogram true | false]
[-connection_type ALL | BLOCK_TO_BLOCK | BLOCK_TO_MACRO | BLOCK_TO_IO]
[-cell_type ALL | BLOCK | HARD_MACRO | EXTRACTED_TIMING_MODEL]
[-pin_type ALL | SIGNAL_PINS | PG_PINS]
[-corner_keep_out true | false]
[-detour_tolerance *tolerance_ratio*]
[-error_view *view_name*]
[-exclude_unplaced_objects true | false]
[-filename *file*]
[-layers true | false]
[-location true | false]
[-missing true | false]
[-nets *net_collection*]
[-net_length *net_length*]
[-off_edge true | false]
[-offset true | false]
[-order true | false]
[-output_directory *directory*]
[-pins *pins*]
[-pin_blockage true | false]
[-pin_density_map true | false]
[-pin_detour true | false]
[-pin_direction true | false]
[-pin_guide true | false]
[-pin_mask_constraint true | false]
[-pin_size true | false]
[-pin_spacing true | false]
[-technology_spacing_rules true | false]
[-ports *ports*]
[-pre_route true | false]
[-report_net_details true | false]
[-routing_corridor true | false]
[-shorts true | false]
[-self]
[-sides true | false]
[-single_pin connected | unconnected | all]

[-stacking true | false]
[-synthesized_pins true | false]
[-wire_track true | false]
[-wide_track true | false]
[-window_length *window_length*]
[*list_of_objects*]

Data Types

tolerance_distance float
file string
tolerance_ratio real
view_name string
net_collection list
net_length real
directory string
pins collection
ports collection
list_of_objects list

ARGUMENTS

-alignment true | false

Checks block pin alignment QoR. The alignment check is limited to two-pin nets with at least one connection to a block (or Hard-Macro/ETM if the appropriate **-cell_type** option is used) and no connection to a standard cell pin. Nets not connected to a block or connected to standard cell pins, or those with more than two connections, are not checked. Nets are not checked if they are connected to blocks that are disabled for planning. When used with the **-self** option the check is limited to two-pin nets with one connection to a top-level terminal (and no connection to standard-cell pins).

If this option is not specified, the command skips alignment checking. Note that two pins are considered unaligned if their pin bounding boxes are mis-aligned by greater than the **-alignment_tolerance_distance**. But if the tolerance option is not used and the bounding boxes of two pins overlap and the extent of the overlap is greater than two-thirds of smaller bounding box then pins are not considered to be unaligned.

On the other hand if the **-alignment_tolerance_distance** is specified/used then that distance is used to compute alignment.

Note that by using the option along with the **-cell_type** option, alignment can be checked between hard macro to hard macro instances or ETM to ETM cells.

-alignment_tolerance_distance *tolerance_distance*

Checks that pins are aligned within the specified *tolerance_distance*. The command only reports misaligned pins when the distance between two pins is greater than specified *tolerance_distance*. The distance is the vertical distance if two pins are on horizontal layers, or the horizontal distance if two pins are on vertical layers. The distance is to be specified in microns. You must use the **-alignment true** option with the **-alignment_tolerance_distance** option. When this option is not specified the default value of zero microns is used.

-layer_mismatch_only true | false

Limits the alignment check to two-pin nets that are aligned but have pins on different layers. This option can be used only with the **-alignment** option. When set to true the summary report, detailed net report and list of violating pins returned will all correspond to only such nets.

The default value of this option is false.

-blocked_only true | false

Limits the alignment check to two-pin nets that are aligned but are blocked by a physical block (or hard macro) between the two pins. This option can be used only with the **-alignment** option. When set to true the summary report, detailed net report and list of violating pins returned will all correspond to only such nets. Note that if the physical block is a hard-macro the net is only considered blocked if there is a routing-blockage (on appropriate layer) inside the hard-macro that overlaps the nets bounding-box/path.

-alignment_report_file *file*

Specifies the name of the file where the detailed net alignment reports are written. This option can only be used with the **-alignment** and **-report_net_details** options.

-alignment_histogram true | false

This option is used to generate a histogram of the mis-aligned nets that are alignable. The histogram is based on mis-aligned distance. The nets are grouped in buckets of 50 microns according to their misaligned distance. The option can be used only with the **-alignment** option. The default value is false.

-connection_type ALL | BLOCK_TO_BLOCK | BLOCK_TO_MACRO | BLOCK_TO_IO

Limits the alignment check to two-pin nets of the specified type. This option can be used only with the **-alignment** and **-single_pin** options.

For the single pin check, specifying this switch limits single pin checking to that subset of interfaces. For example between only blocks and hard-macros. Note that this option cannot be used with the **-cell_type** option.

If this option is not specified, the default is ALL, which runs the check on all the two-pin nets in the design.

-cell_type ALL | BLOCK | HARD_MACRO | EXTRACTED_TIMING_MODEL

Directs the command to check the pins of only the specified type of cells. The default value is BLOCK, which limits the checking only to pins on physical blocks. The option cannot be used in conjunction with the **-self** option.

Pin checking on Hard-Macro and Timing-Model instances is limited only to short, preroute, missing, single-pin , alignment, wire-track and wide-track checks. This option is silently ignored for all other checks and the remaining checks will only run on pins of physical blocks.

When the value specified is ALL, then block ETM and hard-macro pins are checked for appropriate violations.

-pin_type ALL | SIGNAL_PINS | PG_PINS

Directs the command to check the pins of only the specified type, that is either power/ground or signal pins or both.

The default is signal_pins, which limits the checking only to pins on physical blocks connected to signal nets.

A pin is considered a power/ground pin by the checker if it connected to a power/ground net and is itself also of the power/ground type. The **place_pins** command also treats PG nets/pins similarly (refer to its man-page for relevant details).

PG (power/ground) pin checking is limited only to the short, single-pin, missing wire-track, wide-track and pin-size checks. This option is silently ignored for all other checks and the remaining checks will only run on signal pins.

When the value specified is ALL, both signal and PG pins are checked for appropriate violations.

Using this option along with the **-nets** , **-pins** or **-ports** options results in filtering of the specified set (of nets, pins or ports) according to the passed pin type value. That is if pin-type SIGNAL is used along with **-pins** then only those pins from the set of pins that are of type SIGNAL will be used for pin checking.

Note that short and spacing violations of signal pins with power/ground pins is checked by default, that is even when the value for this option is signal_pins. When the value is set to ALL then short violations between two PG pins are also checked. Note that checking for spacing violations between two PG pins is not supported.

-corner_keep_out true | false

Checks for corner keepout violations on the edges of physical blocks. If this option is not specified, the command skips the check.

The default value of this option is false.

-detour_tolerance *tolerance_ratio*

Checks that the net detours are within the specified *tolerance_ratio*. You must use the **-pin_detour true** option with the **-detour_tolerance** option. The command reports only nets with the ratio of the total-bounding box to the standard-cell bounding box that is greater than the specified value. The default *tolerance_ratio* is one.

-error_view *view_name*

Specifies the name of the generated error view. The command saves the error view data to a file in XML format with the specified name. No attachment is created nor saved in to the design database. If the view_name file already exists under the current run directory, the command overwrites the file. If a file extension is specified, it must be ".err", otherwise it will be ignored. If no extension is specified, the filename will be appended with the ".err" extension.

If this option is omitted, the error view data is stored as an attachment to the design database. When the design is closed and saved, the error view data is saved to the design database as well. The saved error view data does not change until the next *check_pin_placement* command.

-exclude_unplaced_objects true | false

This option can be used with the **-alignment** or **-pin_detour** options. When used with the alignment option, it excludes nets that have unplaced connections from the alignment check, when set to true. In this context (when used with alignment check), "Un-Placed" refers to a pin on an unplaced cell.

When used with the detour check it skips unplaced cells and dangling block pins from the detour computation. For details on how such dangling pins are handled by default refer to the **-pin_detour** option description.

By default, this option is false.

-filename *file*

Specifies the name of the file where the reports of pin placement checking are written. Cannot be used in conjunction with *-output_directory* option

-layers true | false

Determines if pins are placed on legal metal layers as specified by the *-allowed_layers* option of the *set_block_pin_constraints* or the *set_individual_pin_constraints* command. The command also checks for violations of any per-edge layer constraints that are specified in the pin-constraint file. For information about the default minimum and maximum metal layers used for pin placement, see the *set_block_pin_constraints* command man page.

For pins that are connected to and covered by a top-level terminal, the check ensures that such pins are on the same layer as that top-level terminal. Individual or block-level layer constraints are not checked for such pins.

If this option is not specified, the command skips the layer check.

The default value of this option is false.

-location true | false

Determines if pins with location constraints are placed on their constrained location. Both on-edge and off-edge pins are checked against their location constraints.

For information about setting location constraints on pins review *set_individual_pin_constraints* command. If this option is not specified, the command skips the location check. The default value of this option is false.

-missing true | false

Reports block pins without a pin shape. By default, this option is true.

-nets ***net_collection***

Limits the pin placement check to the specified collection of nets.

-net_length ***net_length***

Limits the detour check to only those nets whose length is greater than the specified value. The default is zero microns, which means that all nets will be checked for detour. Currently this option can only be used with the **-pin_detour true** option. The units for *net_length* are microns. If routing is present for the net, the total length of the route is used. Otherwise, half the perimeter wire-length (from the bounding box of the net) is used to calculate the length.

-off_edge true | **false**

Determines if part of a pin outline coincides with an edge. A block cell pin is flagged if none of the pin edges coincide with its block edges. If this option is not specified, the tool skips the off edge check.

The default value of this option is false.

-offset true | **false**

Checks if pin-placement violates (in decreasing order of priority) individual, topological or bundle pin offset constraints. If multiple constraints exist on the same pin (say individual and bundle for example) then the constraint with higher priority is given precedence and checked against.

If this option is not specified, the command skips the check. Note that even if the offset option is used without the **-side** option explicitly set to true, the command automatically runs the sides check as well. If certain pin/port placement already violates side constraints, it will automatically skip for offset checking.

If a range is specified (when specifying the offset constraint) then the pin's placement is checked against that range. If a specific offset is specified then the pin should be placed either at that offset or at the closest valid wire-track to that offset.

The default value of this option is false.

-order true | **false**

Checks pin placement for order constraint violations. The order constraints specified in pin-constraint file with the **read_pin_constraints** command are used for this check.

The default value of this option is false.

-output_directory ***directory***

Writes individual report files for each check (layer, offset, side, and so on) to the specified directory. If the directory does not already exist, the command tries to create the directory. If creation is successful and write permissions are appropriate, the command checking output is written to individual files created in this directory instead of the prompt. The files created also have a list of pins associated with the particular violation. The option cannot be used together with the *-filename* option. If you omit the **-output_directory** option and specify the **-filename** option, the command writes a single file that combines the checks.

-pins ***pins***

Limits the pin placement check to the specified collection of pins.

-pin_blockage true | **false**

Checks if a pin overlaps with an associated pin blockage or if the terminal of the pin abuts to or overlaps with a routing blockage, and if the layer of the pin is the same as the layers associated with the blockage. If this option is not specified, the command skips the pin blockage check.

The default value of this option is false.

-pin_detour true | false

Checks for pin-detour violations. The option is used to report pins that might not be optimally placed. The check reports nets that have a bounding-box greater than the bounding box of the standard-cell pins and top-level terminals/ports that it connects.

The check runs on nets that have at least two pins with a minimum of one block pin. This includes nets that connect top-level ports/terminals to block pins. Pins that are unplaced are not included in the bounding box comparison. Also unconnected or dangling block pins are accounted for by projecting the center point of the standard-cell bounding box (corresponding to the nets connections) onto the edge of the dangling block pins. This projected point(s) are added to the standard-cell bounding box to compute a more meaningful detour ratio in the presence of dangling connections.

When this option is used along with the **-self** option, it limits the detour checking to only those nets that have a connection to a top level port in addition to having at least one block pin.

The option **-exclude_unplaced_objects** can also be used along with this option to filter out or skip dangling block pins from the detour computations. When set to true, block pins that have no external connections or those whose external connections are only other dangling block pins are excluded from the detour computations.

When the app option **plan.pins.path_based_pin_placement** is used the detour check reverts to use the register based bounding box instead of standard-cell based one (for qualifying nets). For details on how that bounding box is computed refer to the man-page for that option.

When the app option **plan.pins.detour_supernet_mode** is used the detour check skips over repeaters and considers the flat net during detour computation. Note that the path based option has precedence over the supernet based option. Please refer to the man-page of these options for more details.

Note that a net is skipped for detour checking if all the physical blocks it connects to are disabled.

The default value of this option is false.

-pin_density_map true | false

Checks and reports on the prompt and the GUI the block pin density on the edges of each physical block in the design. The report is per layer and each edge is sub-divided into buckets or windows for density computation. The density on each layer in each bucket is defined as the ratio of the number of pins by the number of available slots/wire-tracks (on that layer) within the bucket/window. The text version of this report is automatically redirected into a new file called block_pin_density.rpt created in the current working directory. If the directory is not editable or the file cannot be created, a warning is issued and the report is instead redirected to the prompt.

The option **-window_length** can be used to control the size of the window. If not specified a default window size as a multiple of site row height is used.

The default value of this option is false.

-pin_direction true | false

Checks for pin direction mismatch violations. It is divided into two categories.

- abutted or stacked pin direction mismatch
- multi-driven or floating nets that connect to hierarchical physical blocks

By abutted or stacked pins, it refers to two pins abutted or stacked by abutting hierarchical physical block edges, and these two pins are connected by the same net. Two abutted pins must be on two hierarchical physical blocks of the same physical hierarchy. Two stacked pins must be on two hierarchical blocks that one is the parent of the other.

Please note, the power, ground, or tie-off nets are not checked for pin direction mismatch.

For a pair of abutted pins, only the following are considered as legal:

- one is input and the other is output, or

- one is inout and the other does not matter

For a pair of stacked pins, only the following are considered as legal:

- both pins are input or both pins are output, or
- one is inout and the other does not matter

For a net that is not multi-driven nor floating, it must satisfy

- it must have a driver from a pin or port of input, output or inout pin direction, and
- if the driver is of input or output direction, there can be only one such driver

The default value of this option is false.

-pin_guide true | **false**

Checks if the center of a pin is outside the associated pin guide and if the layer of a pin is NOT on a layer specified by the pin guide. If this option is not specified, the command skips the pin guide check.

The default value of this option is false.

-pin_mask_constraint true | **false**

Checks if there are any mask_constraint violations on the block pins, such as abutted block pins with conflicting mask_constraint settings, block pin and wire track with conflicting mask_constraint settings, and so on. When checking if the block pin and the wire track have conflicting mask_constraint settings, the tool only checks the wire track owned by the block. If the block does not have a wire track within it, then the tool will check the block pin against top-level wire track instead.

By default, this option is false.

-pin_size true | **false**

Checks if pins honor their width and length constraints. If the pins do not have width or length constraints, but are on the double patterning layers, this option checks if the pins honor the default double patterning pin length requirement. If pins have width and length constraints and are also on double patterning layers, this option checks only if the width and length constraints are honored. If there are non-default routing rules associated with the pin that have size/width constraints, those are checked as well. If a pin is placed on a wire track with non-zero track width, then the command checks to ensure that the pins width is the same as the track width.

If there is a min-area rule specified on the technology file for the layer, then the pins-area is checked against this rule.

It also checks to ensure that the pin-length is a perfect multiple of the grid size (if a grid is present/defined). For detailed explanation of the handling of pin length, width and area constraints, see the man page for the **place_pins**

The default value of this option is false.

-pin_spacing true | **false**

Determines if any of the following object-pairs along the edge (the two objects overlap in the direction perpendicular to the edge), are closer than the number of wire tracks (or distance) specified by the *-pin_spacing* option of the *set_block_pin_constraints* or the *set_individual_pin_constraints* command.

- two signal pins
- a signal pin and a power/ground pin (Note that spacing between two power/ground pins is not checked.)
- a signal pin and a pre-route shape/via

If there are NDR rule constraints associated with the pin (or associated net) then the spacing is checked against those constraints as well. The command also checks for violations of any per-edge layer constraints that are specified in the pin-constraint file.

The report has sub-sections for these different type of constraints. Note that the spacing check between pins is limited to pins on

the same block edge.

If this option is not specified, the pin spacing check is not performed.

The default value of this option is false.

-technology_spacing_rules true | false

Determines if any two signal pins (or a signal pin and power/ground pin) or a signal pin and a pre-route are closer than any of the technology related spacing constraints. The supported spacing rules that are checked are the corresponding layers minimum spacing rule (as defined in technology file), fat-metal spacing rules and basic, color-based span spacing rules that might be defined in technology file.

Note that spacing is only checked for pins/shapes along a block edge. That is off-edge pins are not checked for spacing violations. Also pre-route shapes from inside a hard-macro are not considered when checking spacing.

For two objects to be checked for spacing (except for end to end spacing), they must overlap in the direction perpendicular to the edge. End to end spacing checks the spacing between the ends of two objects (signal pins and signal/PG routing shapes), for objects to qualify for end to end spacing they must overlap in the direction of the edge (without being shorted together). Note that the spacing check between pins is limited to pins on the same block edge.

The report has sub-sections for these different type of rules. Note that spacing between two power/ground pins is not checked and that the **-pin_spacing** option only checks user specified constraints (like block or individual pin spacing constraints) and this option checks only technology related rules/constraints.

The default value of this option is true.

-ports *ports*

Limits the pin placement check to the specified collection of ports.

-pre_route true | false

Determines if any pin is shorted with any preroute (signal or power/ground) or via. The check is used to ensure that no pins overlap with and are on the same layer as a section of preroute. Note that the *-shorts* option does not check for shorts against preroute.

Note that even when used with the **-cell_type HARD_MACRO** option the checker does not load and check against any preroutes that might exist inside the hard-macro. It does however check the hard-macro pins for shorts against any preroutes that are created outside the hard-macro itself.

The default value of this option is false.

-report_net_details true | false

Reports detailed net information on pins with alignment violations. You must use the **-alignment true** option with the **-report_net_details** option. If this option is not specified, the command does not include net details in the report output. When the **alignment_report_file** option is used the report is written to that specified file.

The following definitions are relevant to this report :

Aligned: The distance between the two pins is less than specified *tolerance_distance*. The distance is the vertical distance if two pins are on horizontal layers, or the horizontal distance if two pins are on vertical layers. The default value of *tolerance_distance* is zero microns.

Unaligned: The distance between the two pins is more than the specified *tolerance_distance*.

Blocked: There is a physical block (including hard macros) along the line between the two pins of the net. Note that if the physical block (that lies along the line between the two pins of the net) is a hard-macro the net is only considered blocked if there is a routing-blockage (on appropriate layer) inside the hard-macro that overlaps the nets bounding-box/path.

Unblocked: There isn't a physical block or hard macro along the line between the two pins of the net. Note that if the physical block (that lies along the line between the two pins of the net) is a hard-macro the net is only considered blocked if there is a routing-blockage (on appropriate layer) inside the hard-macro that overlaps the nets bounding-box/path.

Unalignable: The pins of the net are unalignable due to some combination of pin side, pin-blockages, pin-guides corner-keepout constraints and the relative placement of the blocks. Pins of two child blocks that are not on facing edges are considered unalignable. A net that connects a parent block port/pin to a pin on the parents child block is considered alignable if it possible to align the two pins of the net by placing the parent block port/pin on another allowed side.

Alignable: These nets are alignable given the constraints listed above. Most typically two pin nets are considered alignable when the pins are placed on facing edges.

The report is divided into 5 parts: (1) aligned nets on different layers, (2) aligned blocked nets, (3) unaligned alignable nets, unblocked, (4) unaligned alignable nets, blocked, (5) unalignable nets. Each section begins with one of the following commented header lines to indicate the type of unaligned or aligned nets.

- aligned nets on different layers:
- aligned blocked nets:
- unaligned alignable unblocked nets:
- unaligned alignable blocked nets:
- unaligned unalignable nets:

Following the header line, the report lists the details of each aligned or misaligned net. The details include the name of the net and its two pins, the type of the net/connection and whether the net is aligned or misaligned. The report also reports if the net is blocked/unblocked and if either pin is fixed.

The connection is between two pins that are from a block cell or top design (BLOCK), top-level I/O pins (IO), pad cell(PAD) or hard macro(HM). One of the pins must connect to a block cell. The following shows an example connection:

Two-Pin net {NetName} with pins {PinA} and {PinB} of type BLOCK->IO
is aligned. The net is blocked. {PinA} is Fixed.

For nets that are unalignable, the reason that they cannot be aligned is provided.

The default value of this option is false.

-routing_corridor true | false

Checks if the center of a pin is outside the associated routing corridor and if the layer of a pin is NOT on a layer specified by the corridor. If this option is not specified, the command skips this check.

For nets with feedthrough pins, the checking depends on the properties of the routing corridor. If the routing corridor is connected and covers all its associated pins/ports, then each pin (including feedthrough pins) created by pin-placement should be within the routing corridor. If the routing corridor is

not fully connected or does not cover all its associated pins/ports, then at least one of the original or feedthrough pins (per block) should be within the corridor.

In addition, if there is no overlap between a corridor and the block boundary, the pins of that block will not be checked against the corridor. When used with the **-self** option, the command checks all top-level ports against their associated routing corridors.

The default value of this option is false.

-shorts true | false

Checks for shorts that are caused by two pins on the same layer that touch or overlap each other. Pins that are connected to the same net are not considered to be shorted. If this option is not specified, the command performs the shorts check. For ports with

multiple-pins/EEQ-pins, each of the pins are checked for violations. And each individual violation is reported in the error-browser GUI. On the command prompt though any combination of two ports that are shorted are reported a single time (independent of the number of violations their individual pins create).

The default value of this option is true.

-self

Limits the pin placement check to the top level ports.

-sides true | false

Checks for pin side violations that are caused by a pin's placement violating side constraints. The violating side constraint can be any of the following (in order of decreasing priority): individual pin constraints, topological, bundle or block constraints. In the presence of multiple constraints (individual and block constraints for example) the one with the higher priority is given precedence and checked against.

Note that command reports violations and sides with respect to the reference block. The default value of this option is false.

-single_pin connected | unconnected | all

Checks for unroutable pins on abutted edges. Valid values for this option are:

- **connected**
  A block pin is flagged as a single (unroutable) pin if any one of the following are true:
  - The block pin is placed on a block edge that abuts another block

    edge and the block cell pin does not abut the other block pin on the same metal layer, or these two pins mismatch in width.
  - The block pin is placed on a block edge that coincides with the

    top-level cell boundary and connects to anything other than a terminal (top-level pin), unless the pin connects only to other pins on the same block and those pins are stacked on top of the pin.
  - The block pin is placed on a block edge that coincides with the

    top-level cell boundary and the block cell pin is connected only to a terminal, but the terminal is not stacked on top of the block cell pin on the same metal layer, or the terminal and the cell pin mismatch in shapes.
- **unconnected**
  A block cell pin is flagged as a single (unroutable) pin if either one of the following is true:
  - The block cell pin is logically unconnected to any pin or terminal

    and is placed on a part of the block edge that abuts another block edge.
  - The block cell pin is logically unconnected to any pin or terminal

    and is placed on a part of the block edge that coincides with the top-level cell boundary.

  When you specify **-single_pin unconnected**, the command also checks terminals. A terminal is flagged if it is logically unconnected and is located on a top-level cell boundary that abuts a macro edge.
- **all**
  A block cell pin is flagged as a single (unroutable) pin if any of the scenarios described for **connected** or **unconnected** occur.

The above checking applies only to pins that are placed on (abutted) edges. For **off-edge pins** that overlap with other physical blocks above/beneath them, the command checks to see if there is a corresponding matching pin above/below it. It also checks to ensure the matching pin is of the same size/shape and layer as the off-edge pin. Off-edge pins that have routing to other

terminals are skipped for this type of checking.

When checking for single pins the pin being checked is expected to be of the same type (signal or PG) as its matching/corresponding pin (whether abutted or stacked). Also the the abutted pins are expected to be of compatible directions.

For block edges that abut (or overlap in case of parent and child block) hard-macro edges, a pin placed on such an edge is considered legal if it either has a matching hard-macro pin or there is no blockage/routing-shape on the same layer within the abutted hard-macro abutted/adjacent to the port.

Note that **physical-only cells** (such as fillter of cap cells) are ignored by the single pin check. For pins abutted to their parent block pins, the single pin check also reports any cases where the child block pin is shorter than the abutted parent block pin. The opposite scenario (that is child block pin is longer) is not considered a violation.

Additionally the **-connection_type** option can be used to filter/restrict reporting/checking of single pin violations to a subset of edge interfaces.

If this option is not specified, the command skips the single pin check.

-stacking true | false

Checks if there exist any stacking violations. Whether or not a pin to pin stacking or a pin to power or ground preroute stacking is regarded as a violation is determined by the setting of *-stacking_allowed* option of command *set_block_pin_constraints*.

-synthesized_pins true | false

This check reports and returns all the pins inserted by the tool to avoid single pins, during pin-placement on abutted designs. If this option is not specified, the command skips this check.

The default value of this option is false.

-wire_track true | false

Determines if pins are centered on wire tracks. A vertical wire track on the same layer as the pin must coincide with the vertical central axis of the pin for pins on a horizontal edge. A horizontal wire track must coincide with the horizontal central axis of a pin on a vertical edge.

Off-edge pins are checked against the wire-track only if the **-off_edge** option is also set to true along with the **-wire_track** option. For an off-edge pin, the command verifies that off-edge pins are centered on wire tracks in the preferred direction of the metal layer of the pins. The pins are not checked against the wire tracks in the nonpreferred direction. If this option is not specified, the command skips the wire track check.

The default value of this option is false.

-wide_track true | false

Checks that pins satisfy track width constraints. When set to true, the check flags two types of violations: 1) pins that are placed on reserved-for-width tracks, 2) pins whose width is different than the width specified for the track.

The default value of this option is false.

-window_length *window_length*

Specifies the window length to be used during pin-density computation. The value is to be specified in microns, if not specified then the command uses a default value (that is reported on the prompt) for computing the pin density. The default value is a function of site row height.

If the specified window length is less than the default, the tool issues a warning message and uses the default. The default is a multiple of site-row height and is computed based on the design and library technology.

This option can only be used with the **-pin_density_map** option set to true.

list_of_objects

Specifies the collection of block cells for pin placement checking. The command checks pin placement only on the blocks you specify. If no blocks are specified, the command checks the pins on all block cells.

DESCRIPTION

This command performs the pin placement check on specified pins or ports of specified cell type (specified by options **-pins**, **-nets**, **-ports**, **-self**, **-cell_type**). If multiple options are specified, the pin placement check will be performed on the objects of the common sets, i.e. objects met all specified conditions. If none of these options are specified, the pin placement check will be performed on all of the block pins whose parent blocks are enabled.

When performing the pin placement checks, this command uses the constraint values set by the commands **set_individual_pin_constraints**, **set_topological_pin_constraints**, **set_bundle_pin_constraints**, or **set_block_pin_constraints** (these constraints can also be set by **read_pin_constraints** using pin constraints file). The command uses default pin constraints if there are no pin constraints applied.

If there are no options specified for the types of checks, the command checks only for missing pins, short pins and min spacing (specified by the technology file) violations. By default the command examines multiple levels of physical hierarchy (MPH support) for pin-constraint checking. The **set_editability** can be used to enable/disable checking on individual reference-blocks or levels of physical hierarchy. Some options have specific behavior relating to how they support disabled blocks. Please refer to the description under each option to understand the behavior.

EXAMPLES

The following example checks all blocks in the design for shorted and missing pins. It also determines if any two signal pins are closer than the number of wire tracks previously specified by the **-pin_spacing** option of the **set_block_pin_constraints** command.

```
prompt> check_pin_placement -shorts true -missing true -pin_spacing true
```

SEE ALSO

place_pins(2)
report_block_pin_constraints(2)
set_block_pin_constraints(2)
set_editability(2)

check_placement_constraints

Checks placement constraints before running **create_placement** or **place_opt**.

SYNTAX

status **check_placement_constraints**

DESCRIPTION

This command is used to check coarse placement constraints before running **create_placement** or **place_opt**. If any infeasible constraints are found, information about the constraints will be written out for the user to fix.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example checks placement constraints:

```
prompt> check_placement_constraints
```

SEE ALSO

create_placement(2)
place_opt(2)

check_pre_pin_placement

Checks for unfriendly user constraints and design issues that might cause pin placement errors. Run this command before performing pin placement.

SYNTAX

string **check_pre_pin_placement**
[-filename *file_name*]
[-self]

Data Types

file_name string

ARGUMENTS

-filename *file_name*

Writes out the placement check results to the specified file name. By default, the command only writes the pin placement check results to the console.

-self

When specified, check will be performed on top ports, instead of block pins. to the console.

DESCRIPTION

This command issues a warning message for one the following scenarios:

- The allowed-side constraint conflicts with the allowed-layer constraint.
- A edge of a block does not have tracks overlapping with it.
- An off-edge fixed pin has topological constraints.
- A fixed pin is outside of block boundary.
- Any side of the bbox for a block cannot be touched by the boundary for that block.
- Inconsistent boundary for a same block between design/abstract view and frame view.
- A fixed pin has topological constraints and its parent cell is facing the target cell, however the pin is not facing the target cell.

- A fixed pin has blockage, pin guide, route corridor or pin short violations.
- A fixed pin is abutted with another block, and the abutted-edge is close to a pin/routing/metal-blockage. In this situation the abutted pin that is created will overlap with the blockage.
- A topological constraint specifies a path that starts from and ends at the same block, but does not specify which pin the path should start from and end at.
- A topological constraint specifies direct connection between two cells which is impossible in the given floorplan.
- A topological constraint specifies direct connection between a port/pin and its target which is impossible in the given floorplan.
- A topological constraint specifies direct connection between sides for two cells which is impossible in the given floorplan.
- A topological constraint specifies very unfavorable side connection: the sides specified are opposite to the facing edges.
- Two cells can only be connected through other cells, while these other cells are neither allowed for feedthroughs, nor enabled for planning.
- A port/pin can only connect to its target through other cells, while these other cells are neither allowed for feedthroughs, nor enabled for planning.
- Inconsistent blockages across MIBs.
- Range/offset overlap between net/bundle constraints.
- Range/offset overlap between net/bundle constraint and pre-existing fixed pin.
- A net is included in multiple bundles.

EXAMPLES

The following command checks the design for possible issues before performing pin placement with the **place_pins** command.

```
prompt> check_pre_pin_placement
```

The following command checks the design for possible pin placement issues and writes out the list of checks performed and results to the file named

```
prompt> check_pre_pin_placement -filename pinplacement.rpt
```

SEE ALSO

place_pins(2)

check_pre_place_io

Check design data before IO placement and bump assignment.

SYNTAX

status **check_pre_place_io**
[-io_guides]
[-pad_assignment_file *filename*]
[-matching_types]

Data Types

filename string

ARGUMENTS

-io_guides

Check whether there exists any IO guide that cross any block boundary.

-pad_assignment_file *filename*

Check whether user pad assignment is correct.

-matching_types

Check whether the matching types do not have bumps in blocks, MIB violations, netlist violations, has enough bumps and has unique components. The checking for netlist violations is a step by step procedure. For all violating matching type net violations a specific warning will be issued and the user will also be informed if the matching type has to be removed from the design if it has conflicts with a specific net in the design.

DESCRIPTION

This command check for design data errors before user runs place_io command. It will not change the design data.

EXAMPLES

The following example performs design data checking.

```
prompt> check_pre_place_io -io_guides -pad_assignment_file file_name
```

SEE ALSO

place_io(2)

check_pt_qor

Checks PrimeTime QoR such as timing and power.

SYNTAX

status **check_pt_qor**
  -reset

status **check_pt_qor**
  -report_script file_name

status **check_pt_qor**
  [-type global_timing | summary | max_delay | min_delay
      | max_transition | max_capacitance | power | noise]
  [-pba_mode none | path | exhaustive]
  [-group *group_name*]

  [-from *from_list*
    | -rise_from *rise_from_list*
    | -fall_from *fall_from_list*]
  [-to *to_list*
    | -rise_to *rise_to_list*
    | -fall_to *fall_to_list*]
  [-through *through_list*]
  [-rise_through *rise_through_list*]
  [-fall_through *fall_through_list*]
  [-exclude *exclude_list*
    | -rise_exclude *rise_exclude_list*
    | -fall_exclude *fall_exclude_list*]
  [-nworst *paths_per_endpoint*]
  [-max_paths *max_path_count*]
  [-unique_pins]
  [-slack_greater_than *minimum_slack*]
  [-slack_lesser_than *maximum_slack*]
  [-include_hierarchical_pins]
  [-start_end_pair]
  [-path_type *format*]
  [-input_pins]
  [-nets]
  [-nosplit]
  [-transition_time]
  [-capacitance]
  [-significant_digits *digits*]
  [-crosstalk_delta]
  [-derate]
  [-variation]
  [-exceptions dominant | overridden | all]

[-voltage]
[-verbose]
[-save_session *directory_name*]

Data Types

file_name string
directory_name string
from_list list
rise_from_list list
fall_from_list list
to_list list
rise_to_list list
fall_to_list list
through_list list
rise_through_list list
fall_through_list list
exclude_list list
rise_exclude_list list
fall_exclude_list list
paths_per_endpoint integer
max_path_count integer
group_name list
minimum_slack float
maximum_slack float
format string
digits integer

ARGUMENTS

-reset

Specifies to terminate current PrimeTime session.

-report_script *file_name*

Specifies a TCL script file to customize the reporting. when -report_script is specified, the -type and related options are all ignored.

-type global_timing | scenario_timing | summary | max_delay | min_delay | max_transition | max_capacitance | power

Specifies the type of QoR to check. The global_timing type is similar to reporting global timing in PrimeTime. The scenario_timing will report merged QoR and QoR for every scenario. The summary type is similar to reporting QoR summary. The max_delay and min_delay types are similar to the **report_timing** command reporting. The max_transition and max_capacitance types are similar to the **report_constraint** command reporting. The power type is similar to reporting power.

-pba_mode none | path | exhaustive

Specifies one of the following path-based timing analysis modes:

- **none** (the default) - Disables path-based analysis and enables ordinary graph-based analysis. This is the fastest mode.
- **path** - Performs path-based analysis on paths after they have been gathered by graph-based analysis, producing more accurate timing results for those paths. To control the ordering of the paths (either by original graph-based slack or recalculated path-based slack), set the **pba_path_mode_sort_by_gba_slack** variable.

- **exhaustive** - Performs an exhaustive path-based analysis to determine the truly worst-case paths in the design. This is the most accurate and most computation-intensive mode.

-group ***group_name***

Specifies a group of timing paths for timing and nets or cells for power.

Options only available for the max_delay and min_delay types

-from ***from_list***

Reports only paths that start from the specified list of pins, ports, nets, or cell instances; or from startpoints clocked by the named clocks. Valid startpoints are clock pins of sequential devices and input pins of the design.

-rise_from ***rise_from_list***

Same as the **-from** option, except that the path must start with a rising transition.

-fall_from ***fall_from_list***

Same as the **-rise_from** option, except that the path must start with a falling transition.

-to ***to_list***

Reports only paths that end at the specified list of pins, ports, nets, or cell instances; or at endpoints clocked by the named clocks. Valid endpoints are data input pins of sequential devices and output pins of the design.

-rise_to ***rise_to_list***

Same as the **-to** option, except that the path data must end with a rising transition.

-fall_to ***fall_to_list***

Same as the **-rise_to** option, except that the path data must end with a falling transition.

-through ***through_list***

Reports only paths in which the data goes through the named pins, ports, cell instances, or nets. You can use this option multiple times in a command.

-rise_through ***rise_through_list***

Same as the **-through** option, except that it applies only to paths with a rising transition at the specified objects.

-fall_through ***fall_through_list***

Same as the **-through** option, except that it applies only to paths with a falling transition at the specified objects.

-exclude ***exclude_list***

Excludes all paths in which the data goes from, through, or to the specified list of pins, ports, nets, or cell instances.

-rise_exclude ***rise_exclude_list***

Same as the **-exclude** option, but applies only to rising transitions at the named pins, ports, nets, or cell instances.

-fall_exclude ***fall_exclude_list***

Same as the **-exclude** option, but applies only to rising transitions at the named objects.

-nworst ***paths_per_endpoint***

Reports up to the specified number of worst paths per endpoint. The default is 1. A larger value results in a larger report and more

runtime.

-max_paths ***max_path_count***

Reports up to the specified maximum total number of paths among all path groups. The default is equal to the **-nworst** setting, or 1 if the **-nworst** option is not used.

-unique_pins

Reports only the single worst timing path through any given sequence of pins. No other paths are reported for the same sequence of pins from startpoint to endpoint. For example, if the worst path starts with a rising edge at the first pin of a pin sequence, then paths starting with a falling edge are not reported for that sequence of pins.

-slack_greater_than ***minimum_slack***

Reports only paths with slack greater than the specified *minimum_slack* value; these paths have a negative slack better than the specified *minimum_slack* value (or a positive slack that is farther from causing a violation). This option is intended to be used with the **-slack_lesser_than** option to report paths within a specific range of slack.

-slack_lesser_than ***maximum_slack***

Reports only paths with slack less than the specified *maximum_slack* value; these paths have a negative slack worse than the specified *maximum_slack* value (or a positive slack that is closer to causing a violation).

-include_hierarchical_pins

Causes the timing path report to show points for the hierarchical pins crossed, as well as all leaf pins in the path. The hierarchical pins are shown only for documentation purposes; they are reported as having zero incremental delay and do not affect the timing results.

-start_end_pair

Reports the worst path for each startpoint-endpoint pair in the design, including all violating startpoint-endpoint pairs by default, or all startpoint-endpoint pairs with slack worse than the value specified by the **-slack_lesser_than** option if that option is used.

-path_type ***format***

Specifies the types of path information displayed in the timing report, using one of the following format keywords:

- **summary** -- Displays the path startpoint, path endpoint, and slack.
- **end** -- Displays the path endpoint, path delay, required time, and slack.
- **short** -- Displays a report like the default **full** report but omits the intermediate points between the startpoint and endpoint.
- **full** (the default) -- Displays the full data path, including all intermediate points and their incremental and cumulative delay, together with the launch and capture clock arrival times, data required time, and slack.
- **full_clock** -- Displays a **full** report with the addition of the clock path points from the clock source to the launch and capture points.
- **full_clock_expanded** -- Displays a **full_clock** report with the addition of the clock path points between the primary clock source and its related generated clock source point. This report is different from a **full_clock** report only when the clock is a generated clock.

-input_pins

Shows cell input pins as well as cell output pins in the timing path. As a result, the report shows the incremental net and cell delays separately at each point, instead of combined. By default, the report shows only the cell output pins.

-nets

Shows nets in the timing path. By default, the report does not show nets.

-nosplit

Prevents line splitting. This can useful for scripts that extract information from the report. By default, the report generates a new line when the text cannot fit in the alloted space in a column.

-transition_time

Shows the transition time (slew) in the path report for each driver pin and load pin, appearing as an additional column labeled "Trans".

-capacitance

Shows the total capacitance in the path report for each net, appearing as an additional column labeled "Cap".

-significant_digits *digits*

Specifies the number of digits after the decimal point displayed for time values in the generated report.

-crosstalk_delta

Reports crosstalk delta delay values at cell input pins in a column labeled "Delta". The **-input_pins** option is automatically selected. The delta values are computed during crosstalk signal integrity analysis.

-derate

Shows derating factors in a column labeled "Derate".

-variation

Includes parametric on-chip variation (POCV) information in the report in columns labeled "Mean" and "Sensit" if POCV analysis is enabled in PrimeTime.

-exceptions dominant | **overridden** | **all**

Reports user-specified timing exceptions that are satisfied per timing path being reported.

Specifying dominant reports the dominant timing exception in the path. It can be one of the following: false path, minimum and maximum delays, and multicycle paths.

Specifying overridden reports all the timing exceptions that were overridden by the dominant timing exception.

-voltage

Reports the operating voltage for each path element in a column labeled "Voltage", allowing you to debug voltage levels in multivoltage designs.

Options only available for the noise type

-verbose

Reports noise details such as aggressor bump height, width and slack. This option works only with the noise type.

-save_session

Specifies the directory to save PrimeTime session.

DESCRIPTION

When your design is optimized with PrimeTime ECO using the **fix_pt_eco** command, you may want to check QoR in PrimeTime timing values. Use the **check_pt_qor** command to view PrimeTime timing values. The **check_pt_qor** command checks PrimeTime's timing, DRC, and power, and shows their values. By default, the command reports top level timing summary with setup and hold WNS, TNS, and violation count. It is as same as specifying the *-type global_timing*. When the summary type is specified, the command reports QoR summary including setup, hold, max transition, and max capacitance violations. When the max_transition or max_capacitance type is specified, the command reports pins with negative slacks. When the max_delay or min_delay type is specified, the command reports the timing path with its slack values.

When you specify the power type, the command shows leakage, dynamic and total powers in all scenarios.

If you want to customize the reporing, specifify -report_script, which is mutually exclusive with -type.

If you need to terminate PrimeTime, specify the *-reset* option. If the *-reset* option is specified, other options are ignored.

EXAMPLES

Without any options, it shows global setup and hold timing summary, which is similar to PrimeTime's *report_global_timing* command.

```
prompt> check_pt_qor
```

You can view max or min delay timing path in PrimeTime timing by using the -type max_delay or -type min_delay option, which is similar to PrimeTime's report_timing command.

```
prompt> check_pt_qor -type max_delay
```

You can use standard reporting options like -from, -to, -through, etc to specify the timing path that you are interested in.

```
prompt> check_pt_qor -type max_delay -from U1/CK -to U2/D
```

The example below shows max_transition violations.

```
prompt> check_pt_qor -type max_transition
```

For QoR summary, use the -type summary option, which is similar to PrimeTime's report_qor command.

```
prompt> check_pt_qor -type summary
```

For PrimeTime PX power reporting, the -type power option is used as shown below.

```
prompt> check_pt_qor -type power
```

SEE ALSO

set_pt_options(2)
report_pt_options(2)
set_starrc_options(2)

report_starrc_options(2)
eco_opt(2)

check_routability

Checks for many known issues that degrade the performance or quality of routing commands. These checks also include some sanity checks of the input data.

SYNTAX

status **check_routability**
[-access_edge_whole_side true | false]
[-allow_via_rotation true | false]
[-blocked_range *integer*]
[-blocked_range_via_side *integer*]
[-check_congestion_map_overflow *integer*]
[-check_frozen_net_blocked_ports true | false]
[-check_lib_via_cut_blockage true | false]
[-check_min_grid true | false]
[-check_no_net_pins true | false]
[-check_non_standard_cell_blocked_ports true | false]
[-check_out_of_boundary true | false]
[-check_pg_blocked_ports true | false]
[-check_real_metal_blockage_overlap_pin true | false]
[-check_redundant_pg_shapes true | false]
[-check_routing_track_space true | false]
[-check_shield true | false]
[-check_standard_cell_blocked_ports true | false]
[-check_via_cut_blockage true | false]
[-connect_standard_cells_within_pins true | false]
[-error_data *errdata_name*]
[-honor_layer_constraints true | false]
[-obey_access_edges true | false]
[-obey_direction_preference true | false]
[-report_no_access_edge true | false]
[-standard_cell_search_range *integer*]
[-via_ladder true | false]

Data Types

integer integer
errdata_name string

ARGUMENTS

-access_edge_whole_side true | false

Controls the access points when access edges are obeyed (**-obey_access_edges true**).

An access edge is marked by a narrow rectangle, a short line, or a single point. The access edge mark either fully or partially overlaps one of the port edges. When the overlap is not full, such as when the access edge mark is a single point, there are two possible interpretations of the connection rule.

By default (**true**), the connection can be made to any point on the marked port edge. If set to **false**, the connection must be made to the exact access edge mark.

-allow_via_rotation true | false

Controls whether the tool can use rotated vias at pin connections.

By default (**true**), rotated vias are allowed.

-blocked_range *integer*

Specifies the number of pitches for the same-layer accessibility check for macro and top-level ports.

The default is 10.

-blocked_range_via_side *integer*

Specifies the number of pitches for the accessibility check on the other side of an access via.

The default is 10.

This check applies only to macro and top-level ports.

-check_congestion_map_overflow *integer*

Checks global route congestion map overflow larger than or equal to the given value. Please be noted, this check runs a fast global route to get a coarse result. When the user needs an accurate congestion map, please run report_congestion command.

The range of value is 0 to 100. The default value is 30. When the given value is 0, this check is disabled.

-check_frozen_net_blocked_ports true | false

Controls whether the command checks for blocked frozen net ports.

By default (**false**), the command does not check frozen net ports.

-check_lib_via_cut_blockage true | false

Controls whether the command checks for incorrect real metal via cut blockages in the reference libraries.

A via cut blockage is treated as a piece of real metal if all of the following attributes are **false**:

- **icc_route_guide**
- **icc_system_layer_blockage**
- **is_design_rule_blockage**
- **is_boundary_blockage**
- **is_external_boundary_blockage**
- **is_zero_spacing**

When the size of a via cut blockage does not fit a legal cut size (cutWidthTbl, cutHeightTbl, minWidth) specified for the layer in the technology file, it is an incorrect real metal via cut.

In general, the attributes except **is_zero_spacing** are special attributes that are generated from the tool flow. You can set the **is_zero_spacing** attribute to **true** to represent a routing blockage or to **false** to represent a real metal blockage.

An incorrect real metal via cut blockage degrades the routing performance and DRC convergence.

By default (**false**), the command does not check for incorrect real metal via cut blockages in the reference libraries.

-check_min_grid true | false

Controls whether the command checks and reports design objects that are not on the user (manufacturing) minimum grid.

By default (**true**), the command performs the minimum-grid checks.

-check_no_net_pins true | false

Controls whether the command checks the accessibility for pins without an assigned net.

By default (**false**), the command does not check pins without nets.

-check_non_standard_cell_blocked_ports true | false

Controls whether the command checks for blocked macro, top-level, and non-standard-cell ports.

By default (**true**), the macro, top-level, and non-standard-cell ports are checked.

-check_out_of_boundary true | false

Controls whether the command checks and reports the pins that are outside of the design boundary. The design boundary is the same as the die size.

By default (**true**), the command performs the out-of-boundary checks.

-check_pg_blocked_ports true | false

Controls whether the command checks for blocked power and ground ports.

By default (**false**), the command does not check the power and ground ports.

-check_real_metal_blockage_overlap_pin true | false

Controls whether the command checks for real metal routing blockages that touch or overlap a pin inside the reference libraries.

A routing blockage is treated as a piece of real metal when all of the following attributes are **false**:

- **icc_route_guide**
- **icc_system_layer_blockage**
- **is_design_rule_blockage**
- **is_boundary_blockage**
- **is_external_boundary_blockage**
- **is_zero_spacing**

In general, the attributes except **is_zero_spacing** are special attributes that are generated from the tool flow. You can set the **is_zero_spacing** attribute to **true** to represent a routing blockage or to **false** to represent a real metal blockage.

By default (**false**), the command does not check for real metal routing blockages that touch or overlap a pin inside the reference libraries.

When **true**, the command issues a warning message to inform you to review the blockage's attributes.

-check_redundant_pg_shapes true | false

Controls whether the command checks for redundant power and ground shapes that overlap at the same location.

By default (**true**), the command checks for duplicated redundant power and ground shapes.

-check_routing_track_space true | false

Controls whether the command checks for power and ground (PG) via obstacles in the space around each routing track by using the default or nondefault spacing requirement.

By default (**false**), the command does not check for PG via obstacles in the space around each routing track.

When **true**, the command issues a warning message to inform you that these PG vias might degrade the routing quality. The command checks only the PG vias.

-check_shield true | false

Controls whether the command performs checks on the routing shapes related to the shielding flow.

By default (**true**), the command checks the attributes of the routing and shield shapes for known problems in the shielding flow. It does not check the correctness of the shield shapes.

There are three shielding-related checks:

- Non-PG net shape with a **shape_use** attribute of **shield_route**.

  Only PG net shapes can have a **shape_use** attribute of **shield_route**.

- Long PG net wire with a **shape_use** attribute of **detail_route**.

  In general, a PG wire with a **shape_use** attribute of **detail_route** is a tie-off wire. A very long PG wire could be a part of a PG mesh or rail.

- Net with a shield nondefault rule that has a shape whose **shape_use** attribute is not **detail_route**.

For all Zroute commands except **check_routability**, you can disable the shield data checker by setting the **route.common.check_shield** application option to **true**.

-check_standard_cell_blocked_ports true | false

Controls whether the command checks for blocked standard cell ports.

By default (**true**), the standard cell ports are checked.

-check_via_cut_blockage true | false

Controls whether the command checks for incorrect real metal via cut blockages in the design.

A via cut blockage is treated as a piece of real metal if all of the following attributes are **false**:

- **icc_route_guide**
- **icc_system_layer_blockage**
- **is_design_rule_blockage**
- **is_boundary_blockage**
- **is_external_boundary_blockage**
- **is_zero_spacing**

When the size of a via cut blockage does not fit a legal cut size (cutWidthTbl, cutHeightTbl, minWidth) specified for the layer in the technology file, it is an incorrect real metal via cut.

In general, the attributes except **is_zero_spacing** are special attributes that are generated from the tool flow. You can set the **is_zero_spacing** attribute to **true** to represent a routing blockage or to **false** to represent a real metal blockage.

An incorrect real metal via cut blockage degrades the routing performance and DRC convergence.

By default (**true**), the command checks for incorrect real metal via cut blockages in the design.

-connect_standard_cells_within_pins true | false

Controls whether via connections to standard cell pins must be fully inside the pins.

By default (**false**), the via connections do not need to be fully inside the pins.

-error_data *errdata_name*

Specifies the name of the generated error data file.

The default is *check_routability*.err.

If the specified error data file already exists, the command overwrites it.

-honor_layer_constraints true | false

Controls whether the command considers the layer constraints when checking the accessibility of ports and the connectivity of nets.

The layer constraints include the global minimum and maximum routing layer constraints set by the **-min_routing_layer** and **-max_routing_layer** options of the **set_ignored_layers** command; the net-specific minimum and maximum routing layer constraints set by the **-min_routing_layer** and **-max_routing_layer** options of the **set_routing_rule** command; and the freeze layer constraints set by the **route.common.freeze_layer_by_layer_name** and **route.common.freeze_via_to_frozen_layer_by_layer_name** options of the **set_app_options** command.

By default (**true**), the command considers the layer constraints when checking for blocked ports and nets.

-obey_access_edges true | false

Controls whether pins can be connected only at their access edges.

By default (**false**), pins can be connected at any point.

-obey_direction_preference true | false

Controls whether only preferred direction access is allowed.

By default (**false**), access is allowed in both the horizontal and vertical directions. When set to **true**, access is allowed only in the preferred direction.

This option applies only to non-standard-cell ports. Standard-cell ports are usually accessed only by vias. When a wire on the port layer is needed, it can be in any direction.

-report_no_access_edge true | false

Controls whether the command reports the unconnected pins without predefined access edges.

By default (**false**), pins without access edges are not reported.

-standard_cell_search_range *integer*

Specifies the number of pitches for the same-layer accessibility search of standard cell pins.

The default is 2.

-via_ladder true | **false**

Controls whether the command performs the via ladder flow sanity checks.

By default (**false**), the command does not perform the via ladder flow sanity checks. When **true**, the command performs only the via ladder flow sanity checks and does not perform the default checks.

DESCRIPTION

The **check_routability** command checks many issues known to degrade the routing performance or quality, including identification of blocked ports, out-of-boundary ports, minimum manufacturing grid violations, off-routing-track ports, and many miscellaneous sanity checks for routing objects. The command reports the results as part of the standard output and in an error data file. You can display the error data file in the GUI error browser. Some reports are available only in the standard text output and not in the GUI.

The **check_routability** command does not check the routing design rules. To check the routing design rules, use the **check_routes** command.

Most of the issues reported by the **check_routability** command are warnings. Review the generated report to verify that these suspicious issues are intended.

Issues reported as errors should be corrected before running any routing commands.

For legacy reasons, note that some "off-grid" messages mean "off-routing-track".

A logical port can have several physical pins. A port is considered blocked if none of its pins is accessible.

A top-level or macro pin is considered accessible if a legal path can be extended from the pin to a certain distance around it. This path can be on the pin layer or can extend to a neighboring layer by a single via. The required distance on the pin layer and on the layer at the other side of the via are set by the **-blocked_range** and **-blocked_range_via_side** options.

A standard cell pin is considered accessible if there is a legal path that extends to the next layer or a path on the pin layer that extends to the distance set by the **-standard_cell_search_range** option or a shorter path on the pin layer ended by a via to a neighboring layer.

- You can fix blocked ports by removing or modifying the blockages, by moving or modifying the ports, or by modifying the netlist.
- You can fix out-of-boundary ports by moving them or, if appropriate, by fully removing them.
- You can either fix minimum-grid violations or intentionally accept them.

You can run the **check_routability** command at any stage of the routing flow. However, it is most useful when run before global routing. It can be useful to run the command after routing to identify the root cause of certain routing failures.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command performs accessibility checks, out-of-boundary checks, and minimum-grid checks. You should run it before

global routing and, if needed, modify the design according to the generated report.

```
prompt> check_routability
```

The command writes a report to the standard output. The report includes either "no violation" messages or a detailed list of the violations. The following example shows a report for a design with violations:

```
==========================================
==    Check for min-grid violations     ==
==========================================

 ===== Technology min-grid violations =====

 >>> Y minimum grid violation in contact cut via2 20.5544 on layer V2.

 >>>> Found 1 Technology min-grid violations

 ====== Library  min-grid violations ======

 >>> Y minimum grid violation for boundary of library cell AND2X4 at
   (0.0000 -0.0230)_(0.3440 -0.0230).
 >>>> Found 1 Library min-grid violations

 ======= Design min-grid violations =======

 >>> Minimum grid violation in path half width (0.0160) at
   (13.9600 8.7510)_(13.9600 8.7760) on layer M2
   (shape use=DetailRoute, net=vdd).

 >>>> Found 1 Design min-grid violations

>>>>>> Total of 3 min-grid violations

======================================
==  Check for out-of-boundary ports  ==
======================================
>>> Port chip4 clkin at (-0.40 50.500)_(0.20 50.700) on layers M3,M4
  is out of boundary
>>>>>> Found 1 out-of-bound ports

======================================
==     Check for blocked ports      ==
======================================
>>>>>> Port blocked by layer constraints- min/max and freeze layer settings
>>> Port AO222X2_inst_4 pinA at (328.0950 34.4850)_(328.2700 35.9150)
  on layer M1 is blocked by layer constraints(reference=AND2X4, net=A).

>>>>>> Port blocked by check port access

>>> Port AO222X2_inst_4 pinB at (20.4000 40.7500)_(20.4500 40.8500)
  on layer M1 is blocked (reference=AND2X4 , net=A).
>>>>>> The design cannot be cleanly routed because it has 2 completely
  blocked ports

>>>>>> Net blocked by layer constraints- min/max and freeze layer settings
>>> Net netA at (328.0950 34.2050)_(339.300,35.915) is blocked with its
  ports isolated by layer constraints.
```

```
>>> Net netB at (328.0950 34.2050)_(339.300,35.915) is blocked with its
  ports isolated by layer constraints and the shapes on freeze layer
  can't be accessed.

>>>>>> The design cannot be cleanly routed because it has 2 blocked Nets
```

The following command performs only out-of-boundary and minimum-grid checks:

```
prompt> check_routability \
  -check_standard_cell_blocked_ports false \
  -check_non_standard_cell_blocked_ports false
```

The following command does not report minimum-grid violations:

```
prompt> check_routability -check_min_grid false
```

SEE ALSO

create_routing_rule(2)
route_auto(2)
route_detail(2)
route_global(2)
route_group(2)
route_track(2)
set_app_options(2)
set_ignored_layers(2)
set_routing_rule(2)

check_routes

Verifies and reports routing design rule checking (DRC) violations, net opens, antenna rule violations, voltage-area rule violations, and via conversion rates.

SYNTAX

status **check_routes**
[-nets *nets*]
[-open_net true | false]
[-report_all_open_nets true | false]
[-drc true | false]
[-antenna true | false]
[-voltage_area true | false]
[-check_from_user_shapes true | false]
[-check_from_frozen_shapes true | false]
[-coordinates *list_of_regions*]

Data Types

nets collection
list_of_regions list

ARGUMENTS

-nets {*collection_of_nets*}

Specifies the nets to check.

If you use this option, the command checks only for open nets (**-open_net** option), antenna violations (**-antenna** option), and voltage-area violations (**-voltage_area** option) of the specified nets. All other options are ignored.

By default, all nets are verified.

-open_net true | false

Controls whether the command checks for open nets in the block.

By default (**true**), the command checks for open nets.

-report_all_open_nets true | false

Controls whether the command reports all violations when checking open nets.

By default (**false**), the command reports a maximum of 200 open net violations.

-drc true | false

Controls whether the command checks for design rule violations on the specified nets.

By default (**true**), the command checks for design rule violations.

-antenna true | false

Controls whether the command checks for charge-collecting antenna rule violations on the specified nets.

By default (**true**), the command checks for charge-collecting antenna rule violations. Antenna checking requires antenna rules to be defined in the cell library. If no antenna rules are defined, the command skips this check regardless of this option setting.

When a port has multiple physical pins and routing shapes connect to one of the pins to cause antenna violations, antenna violations will not be reported on all of those pins of the port by default. To report violations on all if those pins, use options "-check_from_user_shapes" or "check_from_frozen_shapes".

-voltage_area true | false

Controls whether the command checks for voltage-area rule violations on the specified nets.

By default (**true**), the command checks for voltage-area rule violations.

-check_from_user_shapes true | false

Controls whether the command includes user-created shapes in the checks.

By default (**false**), user-created shapes are not included in the checks.

-check_from_frozen_shapes true | false

Controls whether the command includes frozen shapes in the checks.

Frozen shapes include shapes on frozen layers and shapes of frozen nets.

- A layer is frozen when the **route.common.freeze_layer_by_layer_name** application is set to **true** for the layer.
- A net is frozen when its **physical_status** attribute is set to **locked**. However, the command treats shapes on frozen PG nets that do not have a type of stripe or ring as not-fixed. Shapes on signal nets and of user route type are treated as fixed if option 'check_from_user_shapes' is set to false.

By default (**false**), frozen shapes are not included in the checks.

-coordinates *list_of_regions*

Specifies the rectangular or rectilinear regions in which to perform design rule checking.

Each rectangular region is specified by its lower-left and upper-right coordinates:

{*llx lly*} {*urx ury*}).

Each rectilinear region is specified by its boundary points:

{*x y*} {*x y*} {*x y*} {*x y*} ...)

The coordinate unit is microns. Each region defines a target area for design rule checking. The specified regions can overlap or be disjoint. If you specify more than one area, the command checks the union of all specified areas.

In addition, the command might flag some DRC violations in the neighborhood of the specified regions.

When you specify this option, the following net-based design rules are not checked:

- Open net
- Antenna

- Voltage area
- Tied to rail and tied to rail directly
- Direct connection of secondary power and ground pins and maximum connections for secondary power and ground pins
- Direct connection of comb routing, maximum connections for comb routing, and daisy-chain distance for comb routing

In addition, when you use this option, the command does not report wire length, via count, and redundant via insertion rate.

You cannot use the following net-related options with the **-coordinates** option: **-nets**, **-open_net**, **-voltage_area**, **-report_all_open_nets**, and **-antenna**.

If you do not specify this option, the command checks the entire block.

DESCRIPTION

Verifies, checks, and reports routing statistics for a block. The statistics are based only on the signal routes; PG routes are not included. To report routing statistics that include PG routes, use the **report_design** command.

The **check_routes** command reports the following information for a routed design:

- Open nets
- DRC violations
- Charge-collecting antenna violations

  Antenna analysis is performed only if antenna rules are defined in the cell library.
- Voltage-area violations
- Wire length

  The command reports the total wire length, the total routed wire length, and the wire length per layer.
- Contacts

  The command reports the total number of contacts, the total number of routed contacts, and the number of instances of each via master.
- Wires

  The command reports both the total number of wires and the total number of routed wires.
- Point connections (PtConns)

  A point connection is a wire that has only one connection one each end.
- Redundant via conversion rate

  The command reports the following types of redundant via rates:

  - Total optimized via conversion rate

    The total optimized via conversion rate is computed as

    optimized_via_count / total_number_of_vias

The optimized via count includes both double vias and DFM-friendly bar vias.

The total number of vias (Total Number of Contacts in the generated report) includes both fixed and routed vias. Fixed vias are vias that the router cannot optimize, such as vias at the preroute stage or vias that have a **route_type** attribute of **user_enter**. Routed vias are vias that were routed by the router and can be optimized.

- Total double via conversion rate

  The total double via conversion rate is computed as

  double_via_count / total_number_of_vias

  The double via count includes only the double vias.

- Optimized via conversion rate based on total routed via count

  The optimized via conversion rate based on total routed via count is computed as

  optimized_via_count / number_of_routed_vias

  The number of routed vias (Total Number of Routed Contacts in the generated report) includes only the routed vias.

Note that the via conversion rates reported by the **check_routes** command do not include PG vias. To report the double via conversion rates that include PG vias, use the **report_design -routing** command.

If you specify the **-nets** option, the **check_routes** command checks only for open nets, antenna violations, and voltage-area violations of the specified nets. It does not check for DRC violations, regardless of the setting of the **-drc** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports DRC violations, open nets, antenna violations, and voltage-area violations for all nets in the block.

```
prompt> check_routes
```

The following example checks for open nets, antenna violations, and voltage-area violations, but not for DRC violations.

```
prompt> check_routes -drc false
```

The following example checks for open nets, antenna violations, and voltage-area violations for the N1, N2, and N3 nets. Note that these rules are checked by default, and DRC violations are not checked.

```
prompt> check_routes -nets {N1 N2 N3}
```

The following example reports only antenna and voltage-area violations for the N1 and N2 nets.

```
prompt> check_routes -nets {N1 N2} -open_net false
```

SEE ALSO

route_detail(2)

check_routing_corridors

Checks that all shapes are connected for each routing corridor, and that each corridor contains the pins that connect to the nets associated with the corridor.

SYNTAX

status **check_routing_corridors**
routing_corridor

Data Types

routing_corridor string

ARGUMENTS

routing_corridor

Specifies the name of the routing corridor to be checked. This is a required option. You can specify only one corridor.

DESCRIPTION

This command checks the specified routing corridor for the following properties:

- The shapes that make up the routing corridor are connected
- All pins of the nets associated with the corridor are covered by the routing corridor shapes

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example checks the routing corridor associated with the n7 net.

```
prompt> check_routing_corridors [get_routing_corridors -of_objects [get_nets {n7}]]
Error: Regions in routing corridor 'RC_0' are not connected. (NDMUI-102)
Error: Routing corridor 'RC_0' does not cover pin 'alu/U332/A2'. (NDMUI-103)
```

1

SEE ALSO

add_to_routing_corridor(2)
create_routing_corridor(2)
create_routing_corridor_shape(2)
remove_from_routing_corridor(2)
remove_routing_corridors(2)
report_routing_corridors(2)

check_rp_constraints

Checks relative placement constraints of the specified relative placement groups and reports possible issues, which could lead to critical or non-critical failure.

SYNTAX

collection **check_rp_constraints**
[*rp_group_list*]
[-verbose]
[-*analyze_placement*]
[-region { {llx lly} {urx ury} }]
[-trials number_of_trials]
[-no_pdc]
[-no_adv]
[-threshold threshold]
[-no_rp_constraints]

Data Types

rp_group_list collection
llx float
lly float
urx float
ury float
number_of_trials integer
threshold float

ARGUMENTS

rp_group_list

Specifies the relative placement groups that will be checked. If no any input is provided then all relative placement groups will be checked.

-verbose

Specifies to report exact location of the failures in relative placement group.

-analyze_placement

Specifies that placement analysis will be performed on the relative placement groups. When this option is specified, the other relative placement constraints will not be checked.

-region { {llx lly} {urx ury} }

Specifies the boundary of the region where placement analysis is to be performed. A relative placement group will be considered legal for the sites in this region only if it is placed completely within the region. For an anchored relative placement group, this option will be ignored. This option can be used only along with *-analyze_placement* option.

-trials number_of_trials

Specifies the number of random sites where the placement of relative placement group will be checked. The default value is 100. A value of 0 will force the placement analysis to be performed on all compatible sites. This option can be used only along with *-analyze_placement* option.

-no_pdc

Specifies that no PDC checks are to be performed while doing placement analysis. This option can be used only along with *-analyze_placement* option.

-no_adv

Specifies that no advance rule checks are to be performed while doing placement analysis. This option can be used only along with *-analyze_placement* option.

-threshold threshold

Specifies the threshold pass rate. Relative placement groups with pass rate below the specified threshold value will be reported. This option can be used only along with *-analyze_placement* option.

-no_rp_constraints

When this option is specified, tool will only check if the relative placement group has been successfully placed. The relative placement constraints such as row alignment, column alignment, utilization will not be checked. This option can be used only along with *-analyze_placement* option.

DESCRIPTION

The **check_rp_constraints** command checks relative placement constraints of the specified relative placement groups for possible issues, which could lead to critical or non-critical failures.

The command returns a collection containing the relative placement groups having possible constraint failures. If no constraint failures are seen then an empty string is returned.

When *-analyze_placement* option is provided, the placement analysis of the specified relative placement groups will be performed. For each relative placement group, placement analysis will be done at the random free sites chosen from specified region. For an anchored relative placement group, the placement analysis would be done only at the specified anchor location. The pass rate for a relative placement group is defined as the number of sites at which it could be placed successfully divided by number of free sites at which placement analysis was tried. By default pass rate for all groups is reported, but user could specify threshold to report groups for which pass rate is below threshold. The command returns the collection of reported relative placement groups.

EXAMPLES

The following examples use **check_rp_constraints** to report constraint failures:

prompt> **check_rp_constraints rp_volt1**

```
****************************************************************************
Report : Relative Placement Summary
Total number of specified top level relative placement groups: 1
Total number of relative placement groups which may not honor its constraints: 1
****************************************************************************
RP Group: rp_volt1
----------------------------------------------------------------------------
Warning: The height of relative placement group 'rp_volt1' is more than
the height of voltage area or exclusive move bound. (RPGP-018)

{rp_volt1}
```

prompt> **check_rp_constraints rp_volt2 -verbose**

```
****************************************************************************
Report : Relative Placement Summary
Total number of specified top level relative placement groups: 1
Total number of relative placement groups which may not honor its constraints: 1
****************************************************************************
RP Group: rp_volt2
----------------------------------------------------------------------------
Warning: The bounding box of the anchored relative placement
group 'rp_volt2' is going outside the voltage area or exclusive move bound.
(RPGP-020)
----------------------------------------------------------------------------
Object Type                    Name
----------------------------------------------------------------------------
Exclusive Move Bound                ex_mv_bound1
Voltage Area                   volt1
----------------------------------------------------------------------------

{rp_volt2}
```

The following examples use **check_rp_constraints** to perform placement analysis:

prompt> **check_rp_constraints rp_top_all -analyze_placement**

```
****************************************************************
Report : Placement Analysis Summary
****************************************************************
----------------------------------------------------------------
RP Group                       Pass Rate
----------------------------------------------------------------
rp_top_all                     1
----------------------------------------------------------------
{rp_top_all}
```

prompt> **check_rp_constraints rp_top_all -analyze_placement -trials 200 -threshold 0.3**

```
****************************************************************
Report : Placement Analysis Summary
****************************************************************
----------------------------------------------------------------
RP Group                       Pass Rate
----------------------------------------------------------------
rp_top_all                     0.25
----------------------------------------------------------------
```

{rp_top_all}

SEE ALSO

add_to_rp_group(2)
create_rp_group(2)
remove_rp_groups(2)
report_rp_groups(2)

check_sadp_tracks

Checks the tracks in the design based on self-aligned double patterning (SADP) track rules.

SYNTAX

status **check_sadp_tracks**

ARGUMENTS

The **check_sadp_tracks** command takes no arguments.

DESCRIPTION

This command checks the self-aligned double patterning (SADP) tracks in the current block. The command checks for conflicts within each SADP track set and between different SADP tracks:

- Two tracks should not overlap exactly.
- Spacing between adjacent tracks must be sufficient to ensure SADP rule compliance.

EXAMPLES

The following example checks the SADP tracks in the current block.

```
prompt> check_sadp_tracks
Total Errors = 0
```

SEE ALSO

create_sadp_track_rule(2)
generate_sadp_tracks(2)
remove_sadp_track_rule(2)
report_sadp_track_rule(2)

check_safety_intent

Command for detecting whether the design contains certain types of functional safety intent.

SYNTAX

string **check_safety_intent**
[-verbose]

ARGUMENTS

-verbose

If -verbose is specified, the detected type(s) of functional safety intent is reported.

DESCRIPTION

The command **check_safety_intent** tells whether any functional safety intent (i.e. safety_register_rules, safety_register_groups, repelling_group_bounds) is defined in the current block. When the current block contains such functional safety intent, the returned value is "1". If no such intent is found, the returned value is "0".

This command might be useful when you wish to provide a unified flow-script for both designs with functional safety intent as well as designs without it.

EXAMPLES

The following is an example of the textual report generated by **check_safety_intent -verbose**:

```
prompt> check_safety_intent

Information: Block ChipTop contains FuSa intent (repelling group bounds). (SR-009)
1
prompt>
```

SEE ALSO

report_safety_status
report_safety_register_rules
report_safety_register_groups

check_scan_chain

Allows scan chain structural consistency checking based on the scan chain information stored in the current design.

SYNTAX

status **check_scan_chain**
  [-chain_name *chain_name*]

Data Types

chain_name string

ARGUMENTS

-chain_name *chain_name*

Displays detailed information about the structural checking of the specified scan chain, including location of inconsistency.

DESCRIPTION

This command performs a scan chain structural checking based on the scan chain information stored in the current CEL. If the scan chain passes all the structural checks, then the scan chain status is "VALIDATED"/V and the chain can be physically design-for-test (DFT) optimized. If the scan chain fails the structural checks then the scan chain status is "FAILED"/F and the chain cannot be physically optimized.

By default, the command generates a summary report. To generate a detailed report for a specific scan chain, use the **-chain_name** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example checks for scan chain structural consistency.

```
prompt> check_scan_chain
****************************************
```

```
Report : SCANDEF check
...

****************************************
 Checking between SCANDEF file and design:

 Total SCANCHAINS checked: 6
 VALIDATED :  6
 FAILED    :  0

 Chain name  Status  #cells  #bits  PARTITION   Scan in      Scan out
 268         V       58      58     PLLCKIN_20  i1/DMAX21/Z  i4/reg_1/TI
 269         V       58      58     PLLCKIN_20  i1/DMAX22/Z  i4/reg_2/TI
 270         V       58      58     PLLCKIN_20  i1/DMAX23/Z  i4/reg_3/TI
 271         V       7       5      STCCLK_20   i2/DMAX4/Z   i5/reg_4/TI
 272         V       41      41     CPLCLK_20   i3/DMAX3/Z   i6/reg_5/TI
 273         V       41      41     CPLCLK_20   i3/DMAX9/Z   i6/reg_6/TI
```

The following example checks for the scan chain structural consistency of scan chain 271.

```
prompt> check_scan_chain -chain_name 271
 Checking Scan Chain 271
 STOP: IP2_m_65/IP2_m_75/IP2_m_116/r_276_reg/TI
   thru: IP2_m_65/IP2_m_75/IP2_m_116/r_366_reg/TI
   thru: IP2_m_65/IP2_m_75/IP2_m_116/r_284_reg/TI
   thru: IP2_m_65/IP2_m_75/IP2_m_116/U4/A
   thru: IP2_m_65/IP2_m_75/IP2_m_116/U3/A
   thru: IP2_m_31/IP2_m_210/r_284_reg/TI
   thru: IP2_m_31/IP2_m_210/r_276_reg/TI
   thru: IP2_m_31/IP2_m_210/r_366_reg/TI
 START: IP2_m_31/IP2_m_210/DMAX4/Z

 STATUS:  VALIDATED, Sequential Length = 5, Instance Count = 7,
   Partition = STCCLK_20_20
```

SEE ALSO

read_def(2)
report_scan_chains(2)

check_script

Statically check a script using the a static Tcl syntax checker based on TclPro Procheck and the Synopsys checking extensions for this tool.

SYNTAX

string **snpsTclPro::check_script** [-no_tclchecker_warning]
[-quiet]
[-onepass]
[-verbose]
[-suppress *messageID*]
[-application *appName*]
[-W1]
[-W2]
[-W3]
[-Wall]
filePattern

string *messageID*
string *appName*
string *filePattern*

ARGUMENTS

-no_tclchecker_warning

Don't warn that TclDevKit cannot be found

-quiet

prints minimal error information

-onepass

perform a single pass without checking proc args

-verbose

prints summary information

-suppress *messageIDs*

prevent given messageIDs from being printed

-application *appName*

Check script against application other than this one

-W1

print only error messages

-W2

print only error and usage warnings

-W3

print all errors and warnings

-Wall

print all types of messages (same as W3)

filePattern

file(s) to be checked

USING THE PACKAGE

The **check_script** command is in the snpsTclPro Tcl package, and all of the commands in this package are defined in the namespace snpsTclPro. The way to use this command is to load the package, which will import the commands it exports into the global namespace. This is shown below.

```
shell> package require snpsTclPro
1.0
```

These commands can be added to the .synopsys setup file for the system, user, or current directory, or the commands can be typed when the package is needed.

DESCRIPTION

The **check_script** command simplifies running the TclDevKit tclchecker program. This application is available for purchase from ActiveState (http://www.activestate.com). If the TclDevKit is not available then there is a limited Synopsys Tcl syntax checker that is used by this command. The Synopsys version of this code does not understand Tcl8.4 constructs, and some other checking is limited.

These checkers will check builtin Tcl commands, as well as the Synopsys extension commands which are found in the script. **check_script** will locate the Synopsys checking extensions for the application and will then invoke the checker. The limited checker does not require any external license. It is part of the Synopsys installation.

The **check_script** command will raise a Tcl error if the checker detects errors in the scripts being checked.

The **check_script** command is provided as a convenience to streamline the use of the syntax checker within Synopsys applications. The checker can also be called directly on scripts, without requiring a license for the Synopsys application whose script is being checked. This checker script is available in the auxx/tcllib/bin/check_script in the installation directory of your application. When run standalone the -application option must be specified.

Synopsys Checker Extensions

The Synopsys extensions to the TclPro checker define a number of new warnings and errors. These messages are listed below, along with a short explanation of them.

Error **MisVal**
An option to a command which requires a value is missing its value.

Error **MisReq**
A required option or argument was not specified.

Error **Excl**
Two options which are mutually exclusive were both specified.

Error **ExtraPos**
An extra positional argument was specified to a command.

Error **BadRange**
The value specified for a given argument was outside of the legal range.

Error **UnkCmd**
The command specified is not known by the application.

Error **UnkOpt**
The specified option is not legal for the command.

Error **AmbOpt**
The option specified is not complete and matches 2 or more options for the command.

Warning **NotLit**
This warning indicates that the argument specified to an option was not a literal and therefore could not be checked. This message is suppressed by default. It can be enabled by setting the environment variable SNPS_TCL_NOLITERALWAN to true.

Warning **DupIgn**
An option was specified multiple times to the command, and the first value specified will be the one used. The other values will be ignored.

Warning **DupOver**
An option was specified multiple times to the command, and the last value specified will be the one used. The other values will be ignored.

EXAMPLES

```
shell> check_script myscript.tcl

Loading snps_tcl.pcx...
Loading coreConsultant.pcx...
scanning: /u/user1/myscript.tcl
checking: /u/user1/myscript.tcl
test.tcl:3 (warnVarRef) variable reference used where variable name expected
set $a $b
    ^
test.tcl:5 (SnpsE-MisReq) Missing required positional options for foreach_in_collection: body
foreach_in_collection x {
}
^
test.tcl:9 (SnpsE-BadRange) Value -1 for 'index_collection index' must be >= 0
```

```
index_collection $a -1
child process exited abnormally
Error: Errors found in script.
      Use error_info for more info. (CMD-013)

shell> check_script -Wall anotherScript.tcl

Loading snps_tcl.pcx...
scanning: /u/user1/anotherScript.tcl
checking: /u/user1/anotherScript.tcl
```

CAVEATS

The Synopsys extension messages cannot be suppressed.

Aliases and abbreviated commands will be flagged as undefined procedures.

SEE ALSO

debug_script(2)
package(2)
namespace(2)

check_shapes

Checks and reports all the shapes which meet specified condition in the current block.

SYNTAX

status **check_shapes**
[-uncolored]

ARGUMENTS

-uncolored

Checks and reports uncolored shapes in the current block. A summary of the total count of uncolored shapes found will be printed.

DESCRIPTION

Checks and reports all the shapes which meet specified condition in the current block, the shapes are not only layer shapes but also via shapes.

EXAMPLES

The following example checks and reports uncolored shapes in the current block.

```
prompt> check_shapes -uncolored
****************************************
Report : Reporting uncolored shapes
Design : r4000
Version: P-2019.03-SP3-VAL
Date   : Sun Aug 25 22:46:13 2019
****************************************
Layer  Bbox                              Object
--------------------------------------------------------------------------------
M7     {{76.2300 105.6300} {105.2100 106.0500}}      PATH_37_0
M7     {{30.5900 105.6300} {76.6500 106.0500}}       PATH_37_1
VIA6   {{139.2600 105.6600} {139.6200 106.0200}}     VIA_S_14057
M7     {{139.1800 105.6300} {139.7000 106.0500}}     VIA_S_14057
```

```
***Summary of uncolored shapes***
Number of uncolored shapes found = 4
1
```

SEE ALSO

check_starrc_in_design

validates a configuration file for running In-Design signoff extraction with the StarRC tool.

SYNTAX

status **check_starrc_in_design**
-effort *low* | *medium* | *high*

Data Types

effort string

ARGUMENTS

-effort *low* | *medium* | *high*

- **low**(the default) - to check the settings specified in the config file in set_starrc_in_design for config file path, starrc image path, layer mapping file path, corner mapping, version checks.
- **medium** - to create StarRC command file in Check_StarRC_DesignName sub-directory for users to review or run standalone StarRC, in addition to low effort check.
- **high** - to run StarRC with the generated command file and put the StarRC data in the Check_StarRC_DesignName sub-directory for users to check, in addition to medium effort check.

DESCRIPTION

This command validate the configuration file or run StarRC by using the derived command file with three levels of the validation efforts.

SEE ALSO

set_starrc_in_design(2)
report_starrc_in_design(2)

check_supply_equivalence

Check the supply nets are equivalent.

SYNTAX

status **check_supply_equivalence**
supply_nets
[-functional]
[-pst]
[-verbose]

Data Types

supply_nets list

ARGUMENTS

supply_nets

Specifies the supply nets to be checked. The command fails if the number of supply nets are less than two.

-functional

Specifies that the check is based on functional equivalence. If this options is not specified, the check is based on electrical equivalence by default. This option is mutually exclusive with **-pst** option.

-pst

Specifies that the check is based on PST-equivalence. If this options is not specified, the check is based on electrical equivalence by default. This option is mutually exclusive with **-functional** option.

-verbose

Shows the groups of equivalent supply nets.

DESCRIPTION

The **check_supply_equivalence** command checks whether the specified supply nets are equivalent or not. This command returns 1 if the supply nets are equivalent, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example compares two supply nets that are not electrically equivalent.

```
prompt> check_supply_equivalence {vdda vddb}
 0
```

The following example compares two supply nets that are functional equivalent.

```
prompt> check_supply_equivalence {vdda vddb} -functional
 1
```

The following example compares two supply nets that are pst-equivalent.

```
prompt> check_supply_equivalence {vdda vddb} -pst
 1
```

The following example compares supply nets that are not equivalent and shows the groups of equivalent supply nets.

```
prompt> check_supply_equivalence {vdd1 vdd2 U0/vdd1} -verbose
Information: List of supplies grouped by equivalence - {vdd1 U0/vdd1} {vdd2}. (UPF-183)
 0
```

SEE ALSO

report_supply_nets(2)
report_pst(2)
get_supply_nets(2)

check_targeted_boundary_cells

Checks placement violations of the boundary cells around target objects in the current design.

SYNTAX

status **check_targeted_boundary_cells**
-target_objects {*target_objects*}
[-include_touching_placement_blockages]
[-error_view *view_name*]

Data Types

target_objects	list
view_name	string

ARGUMENTS

-target_objects {*target_objects*}

Specifies the target objects to be around with boundary cells in current design. Objects including macro, voltage area, voltage area shape, placement blockage, route blockage and core area can be specified. This option is a required option.

-include_touching_placement_blockages

Specifies that placement blockages those touch specified target objects should be included. By default, no touching placement blockages are included.

-error_view *view_name*

Specifies the name of the generated error view. Error view data is saved to a file in XML format with the specified name. No attachment is created nor saved into the design database. If the view_name file already exists under current directory, the file will be overwritten. The file extension must be specified as ".err", otherwise it is ignored. If no extension is specified, the ".err" extension is appended to the file name.

DESCRIPTION

This command checks placement violations of boundary cells around target objects according to the boundary cell rules set by *set_boundary_cell_rules*. The violations can be classified as:

1. Continuity Violation:

For this rule, all the boundary cells should be a complete circuit. If no, there will be a violation. If user only insert boundary cells

along all the left and right edges, the boundary cells should cover all the left and right edges. If no, there will be a violation.

2. Corner And Boundary Cell Violation:

This rule requires that all the corner and boundary cells are at the correct location.

3. Orientation Violation:

This rule requires that all the corner and boundary cells are at the correct orientation.

4. Short Edge And Segment Violation:

This rule requires that the length of each horizontal edge of every macro (including hard keepout margin), hard placement blockage, voltage area (including guard band) must be greater than the threshold specified by the option *-min_horizontal_jog*. If not, there will be a violation. This rule also requires that the length of each segment must be greater than the threshold specified by the option *-min_row_width*. If not, there will be a violation.

This command does not check if there is no_1x violation and does not honor the following 4 options of the **set_boundary_cell_rules** command:

-tap_distance

-prefix

-separator

-do_not_swap_top_and_bottom_inside_corner_cell

EXAMPLES

The following example checks for continuity violations, correct corner and boundary cell violations or correct orientation violation in current targeted boundary cells placement.

```
prompt> set_boundary_cell_rules -left_boundary_cell myLib/CellLeft \
        -right_boundary_cell myLib/CellRight \
        -top_boundary_cells myLib/CellTop \
        -bottom_boundary_cells myLib/CellBottom
prompt> check_targeted_boundary_cells -target_objects {PB_7 PB_8}
```

SEE ALSO

check_boundary_cells(2)
compile_boundary_cells(2)
compile_targeted_boundary_cells(2)
remove_boundary_cell_rules(2)
report_boundary_cell_rules(2)
set_boundary_cell_rules(2)

check_tcd_cells

Checks TCD cell placement in the current design and writes out results to the Error Browser.

SYNTAX

status **check_tcd_cells**
-lib_cells *lib_cells*
-window_size *{width height}*
-window_step *{x y}*
[-include_small_windows]

Data Types

lib_cells list of lib cells
width float
height float
x float
y float

ARGUMENTS

-lib_cells *lib_cells*

Specifies the TCD library cells.

-window_size *{width height}*

Specifies the window size width and height.

-window_step *{x y}*

Specifies the x- and y-increment for the next window.

-include_small_windows

Includes small windows at the last row and column for TCD checking. By default, the command excludes small windows.

DESCRIPTION

This command checks if TCD cells exists in each window. You can use the Error Browser to check the results. Each window is the size specified by the **-window_size** option. The first window starts at the lower-left corner of the core. The next window moves both horizontally and vertically by the step specified by the **-window_step** option. If the window step is smaller than the window size,

there will be overlaps between adjacent windows. Overlaps ensure that there is a TCD cell within the window size area.

For example, if the core bounding box has corners at (0,0) and (100,100), the window size is (50,50), the window step is (25,25), and the **-include_small_windows** option is specified, there will be a total of 16 windows as follows:

(0,0) - (50,50)
(25,0) - (75,50)
(50,0) - (100,50)
(75,0) - (100,50)
(0,25) - (50,75)
(25,25) - (75,75)
(50,25) - (100,75)
(75,25) - (100,75)
(0,50) - (50,100)
(25,50) - (75,100)
(50,50) - (100,100)
(75,50) - (100,100)
(0,75) - (50,100)
(25,75) - (75,100)
(50,75) - (100,100)
(75,75) - (100,100)

EXAMPLES

The following example checks if one TCD cell named DM1 exists in each window of size (50,50) and window step of (25,25), including small windows.

```
prompt> check_tcd_cells -lib_cells { DM1} -window_size {50 50} \
  -window_step {25 25} -include_small_windows
```

SEE ALSO

create_backend_tcd_cells(2)
create_frontend_tcd_cells(2)

check_timing

Checks for possible timing problems in the current design.

SYNTAX

status **check_timing**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-all]
[-include *check_list*]
[-exclude *check_list*]
[-override_defaults *check_list*]

Data Types

check_list list
mode_list list
corner_list list
scenario_list list

ARGUMENTS

-modes *mode_list*

Specifies the modes for which to do the timing checks. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is checked separately. If this option is not given, the command does checks for every scenario that is active with the specified corner(s). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to do the timing checks. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is checked separately. If this option is not given, the command does checks for every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios for which to do the timing checks. Each of the specified scenarios is checked separately. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command does checks for all active scenarios and all modes of the design.

-include *check_list*

Adds the checks listed in *check_list* to the default checks. It is an error to give this option with the **-all** option.

-exclude *check_list*

Subtracts the checks listed in *check_list* from the default checks.

-override_defaults *check_list*

Overrides the default checks by using the argument specified in *check_list*. It is an error to give this option with the **-all** option. The options **-include**, **-exclude** and **-override_defaults** can be used together, the precedence is **-exclude > -include > -override_defaults**.

-all

Performs all the checks listed in the following *check_list*. It is an error to give this option with the **-include** or the **-override_defaults** option.

check_list

Specifies the list of checks to be performed. Each element in this list is one of the following strings:

- clock_crossing
- data_check
- gated_clock
- generated_clock
- loops
- multiple_clock
- no_clock
- no_input_delay
- unconstrained_endpoints

DESCRIPTION

This command checks the timing attributes placed on the current design and issues warning messages as needed. The messages provide information that identifies and corrects potential errors. The warnings do not necessarily indicate design problems.

This command performs the default checks when no options are specified:

- gated_clock
- generated_clock
- loops
- no_clock
- no_input_delay
- unconstrained_endpoints

The alphabetically ordered list below shows the meaning of each check:

clock_crossing

Checks clock interactions when there are multiple clock domains. If a clock launches one or more paths that are captured by

other clocks, it will have an entry in the clock crossing report.

- If the two clocks are exclusive or asynchronous clocks, the path is marked with an asterisk (*).
- If the two clocks are set false path (**set_false_path** from clk1 to clk2), the path is marked with a pound sign (#).
- If all the paths between the two clocks are set as false paths, the path is not reported.

data_check

Warns if no clocked signal or multiple clocked signals reach a data check register reference pin. The analysis is done separately for each of the clocked domains.

gated_clock

Warns if the gated clock doesn't reach any register CP pin. The check only performs for ICG cells.

generated_clock

Checks the generated clock network. Five types of issues are reported:

- The definition point of the generated clock has no path to the source point.
- The generated clocks form a loop.
- The specified edge mismatch
- The generated clock is unexpanded
- The generated clock without clock period.

loops

Warns of combinational feedback loops. If the feedback loop is not broken by the **set_disable_timing** command, it is automatically broken by disabling one or more timing arcs.

multiple_clock

Issues a info when multiple clocks reach a register clock pin.

no_clock

Warns if no clock reaches a register clock pin. In this case, no setup or hold checks are performed on data pins related to that clock pin, and the path starting at the clock pin is not relative to a clock.

no_input_delay

Warns if no clock-related delay is specified on an input port, where it propagates to a clocked latch or output port.

unconstrained_endpoints

Warns about unconstrained timing endpoints. This warning identifies the reason of the unconstrained endpoints:

USAGE

There are two ways to run the **check_timing** command:

1. Use the **check_timing** command directly.

   ```
   prompt> check_timing
   Warning: The endpoint 'out1' is unconstrained by the reason of 'false path'.(TCK-001)
   ```

2. Use in the **check_design** flow.

The **check_timing** command is integrated into the **check_design** command as "timing" checks. In the **check_design** flow, all the messages are organized by EMS.

```
prompt> check_design -checks timing
```

This command triggers the **check_timing** command in the **check_design** flow and stores all the messages to the EMS database.

You can redefine the **check_design** checks by using the **create_check_design_strategy** command. For example, to redefine the **timing** check so that it runs the **check_timing** command with the **-include** option, use the following commands:

```
prompt> create_check_design_strategy \
  -define_check timing {check_timing -include {xx}}
prompt> check_design -checks timing
```

Multicorner-Multimode Support

By default, this command works on all active scenarios. To specify different scenarios, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example does default checks without any options:

```
prompt> check_timing
Warning: The register clock pin 'ff1/CP' has no fanin clocks. (TCK-002)
Warning: The endpoint 'ff1/D' is unconstrained by the reason
  of 'unclocked'. (TCK-001)
Warning: A loop 'u4/A u4/Z u3/A u3/Z' is detected, the arc 'u3/A to u3/Z'
  is disabled to break the loop. (TCK-011)
```

The following example adds the clock_crossing check:

```
prompt> check_timing -include {clock_crossing}
Warning: The register clock pin 'ff1/CP' has no fanin clocks. (TCK-002)
Warning: The endpoint 'ff1/D' is unconstrained by the reason
  of 'unclocked'. (TCK-001)
Warning: The register launch clock 'clk1' and capture clock 'clk3'
  are from different domains. (TCK-010)
Warning: A loop 'u4/A u4/Z u3/A u3/Z' is detected, the arc 'u3/A to u3/Z'
  is disabled to break the loop. (TCK-011)
```

The following example removes the loops check:

```
prompt> check_timing -exclude {loops}
Warning: The register clock pin 'ff1/CP' has no fanin clocks. (TCK-002)
Warning: The endpoint 'ff1/D' is unconstrained by the reason
  of 'unclocked'. (TCK-001)
```

The following example adds the clock_crossing check and removes the loops check:

```
prompt> check_timing -include {clock_crossing} -exclude {loops}
Warning: The register clock pin 'ff1/CP' has no fanin clocks. (TCK-002)
Warning: The endpoint 'ff1/D' is unconstrained by the reason
  of 'unclocked'. (TCK-001)
```

```
Warning: The register launch clock 'clk1' and capture clock 'clk3' are
  from different domains. (TCK-010)
```

The following example only does the **unconstrained_endpoints** check:

```
prompt> check_timing -override_defaults {unconstrained_endpoints}
Warning: The endpoint 'ff1/D' is unconstrained by the reason
  of 'unclocked'. (TCK-001)
```

SEE ALSO

check_design(2)
create_check_design_strategy(2)
create_clock(2)
report_constraints(2)
report_timing(2)
set_case_analysis(2)
set_clock_groups(2)
set_disable_timing(2)
set_false_path(2)
set_max_delay(2)
set_output_delay(2)

check_topology_plans

Checks one or more topology plans during different stages of implementation. This command performs a sanity check on base topology plans or checks the quality of results after the topology plan is optimized.

SYNTAX

string **check_topology_plans**
[-filename *file_name*]
[-error_view *view_name*]
[-mode *check_mode*]
plan_list

Data Types

file_name string
view_name string
check_mode string
plan_list list

ARGUMENTS

plan_list

Specifies the topology plans to check. By default, checking is performed on all topology plans in the design.

-filename *file_name*

Writes out the check results to the specified file name. By default, the command writes the check results to the console.

-error_view *view_name*

Specifies the name of the generated error view. The command saves the error view data to a file in XML format with the specified name. No attachment is created nor saved in to the design database. If the view_name file already exists under the current run directory, the command overwrites the file. If a file extension is specified, it must be ".err", otherwise it will be ignored. If no extension is specified, the command appends ".err" to the specified filename.

-mode *check_mode*

Specifies the mode of the check. The following modes are supported:

pre_optimization : Performs a sanity check of the topology plans before optimization

post_optimization : Checks the quality of the topology plan optimization results

pre_implementation : Performs a sanity check of the topology plans before implementation

post_implementation : Checks the quality of the topology plan implementation results

EXAMPLES

The following example checks the current topology plan after performing optimization with the **optimize_topology_plans** command.

```
prompt> check_topology_plans -mode post_optimization [current_topology_plan]
No violation has been found.
```

SEE ALSO

characterize_topology_plans(2)
optimize_topology_plans(2)

check_vclp_design

Used to invoke and perform verification using VC LP.

SYNTAX

status **check_vclp_design**
[-golden_upf_files]
[-write_verilog_options]
[-save_upf_options]

Data Types

-golden_upf_files string
-write_verilog_options string
-save_upf_options string

ARGUMENTS

-golden_upf_files

If FC/ICC2 has Golden UPF Mode enabled, VC LP can be invoked only in Golden UPF Mode and this command option is mandatory. This option can take a list of one or more Golden upf file paths. The file paths must be enclosed in "". The supplemental UPF along with the name map file will be automatically written out by check_vclp_design and VCLP will be run in golden UPF mode.

-write_verilog_options

This option is used to specify additional command options to the write_verilog command that will be invoked under the hood in check_vclp_design.

-save_upf_options

This option is used to specify additional command options to the save_upf command that will be invoked under the hood in check_vclp_design.

DESCRIPTION

The **check_vclp_design** command is used to invoke VCLP from within ICC2/Fusion Compiler and perform verification using VCLP. The VC_STATIC_HOME environment variable must be set, prior to the invocation of this command. To read the design and perform the checks, VCLP sources a template file whose parameters are set up by **check_vclp_design**. Unless users specifically set up the template file by setting the environment variable ICC2_VCLP_TEMPLATE, the template file used by VCLP will be sourced based on

the VC_STATIC_HOME setting. The default template is sufficient for most users.

Following is a list of actions the **check_vclp_design** command performs:

- Opens up VCLP (vc_static_shell) from inside ICC2/Fusion Compiler.
- Performs all the initializations needed for VCLP including setting up the required parameters for VCLP Template file.
- Detects the reference libraries used in the ICC2/Fusion Compiler Run (.nlib files) and extracts the path for the corresponding .db files from the .nlib files and feeds the .db files to VCLP
- Invocation of this command triggers a full run of VCLP. (Verilog netlist and UPF are written out internally and passed as input to VCLP)
- Keeps a communication channel open with VCLP so that users can interact with VCLP with other commands such as **vclp_exec_cmd** or **vclp_zoom_highlight**.

The default write_verilog/save_upf written out internally by check_vclp_design is as below: save_upf -exclude {corner_cells cover_cells end_cap_cells filler_cells pad_spacer_cells physical_only_cells} write_verilog -exclude {leaf_module_declarations corner_cells cover_cells end_cap_cells filler_cells pad_spacer_cells physical_only_cells} -hierarchy all

For designs with block abstracts, ETM, Budget shell, Blackboxes; users will need to edit the template file and point to the new template file with the ICC2_VCLP_TEMPLATE environment variable.

The -golden_upf_files option is mandatory when Golden UPF Mode is enabled. The -write_verilog_options and -save_upf_options command options can only be used in combination and not separately.

SEE ALSO

check_mv_design(2)
vclp_exec_cmd(2)

clock_opt

Synthesize and route the clocks in the current design and then further optimize the design based on the propagated clock latencies.

SYNTAX

status **clock_opt**
[-list_only]
[-from build_clock | route_clock | final_opto | global_route_opt]
[-to build_clock | route_clock | final_opto | global_route_opt]

ARGUMENTS

-list_only

This switch prints out the major stages that constitute the default flow for clock_opt. These stages are: (i) build_clock, (ii) route_clock, (iii) final_opto. Note that the "-list_only" switch cannot be combined with either of the "-from" or "-to" switches.

-from build_clock | **route_clock** | **final_opto**

Specifies either build_clock, route_clock or final_opto as the first stage of the **clock_opt** command.

By default, the flow begins with the build_clock stage if the **-from** option is not specified.

-to build_clock | **route_clock** | **final_opto**

Specifies either build_clock, route_clock or final_opto as the last stage of the **clock_opt** command.

By default, the flow ends with the final_opto stage if the **-to** switch is not specified.

DESCRIPTION

The default behavior of this command is as follows:

(1) Synthesizes and optimizes the clock trees.

(2) Completes the detailed routing of the clock trees.

(3) Further optimizes the design for timing,electrical DRC violations, area, power, and routability, based on actual propagated clock latencies, and legalizes the design placement at the end.

If GR based optimization has been enabled, the clock_opt command will additionally do global_routing, followed by GR-based optimization, and the output will be a design with fully updated global routes and timing based on global routes. Whenever GR based optimization has been run within clock_opt, the subsequent signal routing flow will automatically skip global routing.

If final_opto step is executed after global_route_opt step, it will remove routes before continuing with final_opto step.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example synthesizes, optimizes and routes the clock trees in the current design, followed by further circuit optimization on the design and ends up with an optimized design with a legalized placement:

```
prompt> clock_opt
```

The following example takes in a placed design with synthesized clock trees as its input and first routes the clock trees, then optimizes the circuits in the design for timing, electrical DRC violations, power, area, and routability, and ends up with an optimized design with a legalized placement:

```
prompt> clock_opt -from route_clock -to final_opto
```

The following example takes in a design which has already been run through the clock_opt flow up through the final_opto stage and runs just the global route (GR) based optimization stage (subject to being enabled via the clock_opt.flow.enable_global_route_opt app_option) . The output is a design with global routes and timing based on the global routes.

```
prompt> clock_opt -from global_route_opt
```

SEE ALSO

place_opt(2)
refine_opt(2)

clone_macro_group_packing

Clone macro group packing.

SYNTAX

status **clone_macro_group_packing**
-source source_macro_group
-destinations list_of_macro_groups

Data Types

list_of_macro_groups collection of macro groups

ARGUMENTS

-source source_macro_group

Specifies source macro group.

-destinations list_of_macro_groups

Specifies destinaiton macro groups.

DESCRIPTION

This command copis macro grouping packing from source macro group to destination macro groups. The command will go over each macro group specified in the destination macro groups and use following steps to copy it to a destination macro group.

\n+[step]. If the number of macros of source macro group and destination macro group are not the same, it will not copy the packing from source to desitination and skip it.
\n+[step]. Sorting macros of source macro group and destinaiton macro group based on macro names. Each macro in the source macro list corresponds to the macro of destinaiton macro list at the same location in the list.
\n+[step]. Comparing corresponding macros from these two macro list one by one. If any two macros have different size, the packing cannot be copied and the the destination group is skipped.
\n+[step]. Copy the contour of source macro group to detination macro group in an approriate location.
\n+[step]. Copy location of macro in source macro group to its corresponding macro in the destination macro group. These two macros will have same

relative locations to their contours.

EXAMPLES

The following command copy packing of macro group G1 to destination G2 and G3.

prompt> **clone_macro_group_packings** -source G1 -destination {G2 G3}

SEE ALSO

create_macro_group(2)
clone_macro_group_packing(2)
get_macro_group_packing_clone_candidates(2)

close_blocks

Closes one or more opened block, releasing them from memory when it is no longer used.

SYNTAX

status **close_blocks**
[-save]
[-force]
[-purge]
[*blocks*]

Data Types

blocks collection

ARGUMENTS

-save

This option is used to save the block before closing it (if necessary). This will prevent a failure in the event a close would cause a purge and the block has been modified but not saved.

-force

This option is used to force the block to be closed regardless of whether or not it has been saved. It can be used instead of the **-save** option to force the close but without affecting a save for an open and modified block.

-purge

This option is used to force block from memory, regardless of how many times it has been opened. This seems unsafe but is not since all applications note when blocks are removed from memory. It can be somewhat drastic, so it should be used with care.

blocks

A block name with optional library, label, and view specifications, or a collection of blocks. If the library specification is not present, the **current_lib** is used. If the label specification is not present, the default un-named label is used. If the view specification is not present, then *design view* is used. If the option is not given, the **current_block** is used.

DESCRIPTION

This command decrements and open block's open count and removes one or more block from memory when the reference count drops to 0 (or if the **-purge** option is used). It will fail if the block has been modified but not yet saved, and reference count would

drop to 0 causing the block to be removed from memory, and the **-save** or **-force** option are not used. In the case where the **-save** option is used, the block is saved (only if necessary) before the reference count is decremented. If the **-force** option is used, the command proceeds regardless of the modify status. Please use this option carefully as it is possible to lose data. The **-purge** option is used to force a removal of the block from memory, regardless of its reference count. It will fail if the block has been modified but not yet saved, unless the **-save** or **-force** option is used also.

This command returns the number of blocks removed and 0 if not. If an illegal name(with a slash) is given for the block, a TCL error is raised.

EXAMPLES

This example closes the current block.

```
prompt> close_blocks
1
```

This example forces closed the block Mid, which is being used in Top. This has two effects. First, the instances in Top of Mid become unbound. Second, the block Bot will also be purged since its reference count is decremented when Mid is closed.

```
prompt> close_blocks Mid
1
```

This example closes the block Top, saving it in the process.

```
prompt> close_blocks -save Top
1
```

This example closes a modified and unsaved block that has been opened twice. The first close_blocks only decrements the reference to 1 but doesn't remove the block from memory yet. The second close_blocks would cause an error because the block hasn't been saved:

```
prompt> create_block Top
{top_lib:Top.design}

prompt> open_block Top
Information: Incrementing open_count of block 'top_lib:Top.design' to 2.
{top_lib:Top.design}

prompt> close_blocks Top
Information: Decrementing open_count of block 'top_lib:Top.design' to 1.
1

prompt> close_blocks Top
Error: close_blocks failed. Design 'Top.design' in library 'top_lib' is modified.
```

SEE ALSO

open_block(2)
create_block(2)
copy_block(2)
current_block(2)

current_design(2)
get_blocks(2)
get_designs(2)
move_block(2)
open_lib(2)
remove_blocks(2)
reopen_block(2)
save_block(2)

close_drc_error_data

Closes a physical DRC error data collection.

SYNTAX

status **close_drc_error_data**
drc_error_data
[-save]
[-force]

Data Types

drc_error_data collection

ARGUMENTS

drc_error_data

Specifies the collection of physical DRC error data objects to close.

-save

Saves the pending changes in the exported physical DRC error data before closing the error data file.

This option applies only to external error data files. If the error data file is attached to the current block, use the **save_block** command to save the error data file.

The **-save** and **-force** options, are mutually exclusive; you can specify only one.

-force

Closes the specified collection of physical DRC error data, discarding any unsaved pending changes.

The **-save** and **-force** options, are mutually exclusive; you can specify only one.

DESCRIPTION

The **close_drc_error_data** command decrements the open count of the specified error data collection.

The tool increments the open count for an error data file each time you run the **open_drc_error_data** or **create_drc_error_data** command for the error data file. The tool decrements the open count for an error data file each time you run the **close_drc_error_data** command. The error data collection is removed from memory when the open count reaches zero.

If the error data has unsaved pending changes, the command fails unless you use the **-save** or **-force** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example opens the external physical DRC error data file named my_design_dppinassgn.err, and then closes it.

```
prompt> set data [open_drc_error_data -file_name my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
prompt> close_drc_error_data $data
1
```

SEE ALSO

save_block(2)
write_drc_error_data(2)
open_drc_error_data(2)
save_drc_error_data(2)
get_drc_error_data(2)
create_drc_error_data(2)
remove_drc_error_data(2)
collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

close_ems_databases

Closes the EMS databases in memory.

SYNTAX

status **close_ems_databases**
[-save]
[*ems_databases*]

Data Types

ems_databases list

ARGUMENTS

-save

Writes the currently opened EMS databases to disk before closing them. By default, the EMS databases are not written to disk by this command.

ems_databases

List or collection of EMS databases to be closed.

DESCRIPTION

This command closes the current EMS database in memory, if no argument is specified. If a list or collection of EMS databases is provided as an argument, this command closes only the specified EMS databases, if they are open. The **-save** option writes the EMS databases to disk before closing them.

EXAMPLES

The following example closes the current EMS database, if present.

prompt> **close_ems_databases**

The following example writes out the current EMS database to disk, then closes it.

```
prompt> close_ems_databases -save
```

The following example closes all currently opened EMS databases.

```
prompt> close_ems_databases [get_ems_databases]
```

The following example saves and closes all currently opened EMS databases.

```
prompt> close_ems_databases -save [get_ems_databases]
```

SEE ALSO

create_ems_database(2)
get_current_ems_database(2)
get_ems_databases(2)
open_ems_database(2)
save_ems_database(2)
set_current_ems_database(2)

close_lib

Decrements the open count of a library. A library is removed from memory when its open count is zero.

SYNTAX

status **close_lib**
[-save_designs]
[-force]
[-purge]
[-all]
[-compress]
[-auto_saved_data keep | clear]
[*library*]

Data Types

library collection

ARGUMENTS

-save_designs

Saves any designs in the design library that are open and modified but not yet saved. If the design library contains a technology section that has been modified, it is also saved.

When you use this option, the modified designs and technology section are saved regardless of whether the design library is removed from memory.

This option is not supported in the library manager.

-compress

To be used with "-save_designs" to save libraries in compressed format. This option is applicable to design library only. It is error to use this option without -save_designs.

-force

Removes the library from memory, even if its technology section or some of its designs have been modified but not saved (and you did not use the **-save_designs** option).

-purge

Removes the library from memory, regardless of its open count.

-all

Removes all libraries from memory, regardless of their open counts.

-auto_saved_data keep | clear

Specifies whether to clear or keep auto_saved data of the library. By default, auto-saved data will be kept.

library

Specifies the library to close. This can either be a name or a collection of a single library.

By default, the command closes the current library.

DESCRIPTION

This command decrements the open count of the specified library. The command closes the library and removes it from memory when at least one of the following conditions is satisfied:

- The open count is zero.
- The **-purge** option is used.
- The **-force** option is used.

If you try to remove a library from memory that is on the reference library list of another open design library, the tool issues an error message and does not remove the library from memory.

In the library manager, you can use this command to remove a library from an aggregate library workspace.

EXAMPLES

The following example removes the current design library from memory after saving saving any open and modified-but-not-saved designs.

```
prompt> close_lib -purge -save_designs
1
```

The following example saves the open and modified-but-not-saved designs in all open design libraries and then removes the design libraries from memory.

```
prompt> close_lib -all -save_designs
1
```

SEE ALSO

create_lib(2)
open_lib(2)
save_lib(2)
copy_lib(2)
move_lib(2)

current_lib(2)
get_libs(2)
set_ref_libs(2)
search_path(3)

close_rail_result

Closes the current rail result.

SYNTAX

status **close_rail_result**

DESCRIPTION

This command closes the current rail result.

After the rail result is closed, you cannot access any rail analysis result information such as taps, voltage drop attributes, and power attributes.

EXAMPLE

The following example closes the current rail result.

```
prompt> open_rail_result static
prompt> get_attribute [ get_cell U35 ] static_power
1.0876e-06
prompt> close_rail_result
1
```

SEE ALSO

analyze_rail(2)
open_rail_result(2)
report_rail_result(2)

collection_to_list

Format collection contents as a Tcl list or Scheme string

SYNTAX

list **collection_to_list**
[-name_only]
[-no_braces]
[-brace_with_quotes]
[-newline]
[-truncate *elem_count*]
[-no_sort]
-objects *object_list*
obj_spec_list

Data Types

elem_count int
object_list collection
obj_spec_list collection

ARGUMENTS

-name_only

Prints the object names without specifying the object types.

-no_braces

Prints the output list without beginning and closing curly braces.

-brace_with_quotes

Replaces beginning and closing curly braces with quotes.

-newline

Print each object on a new line.

-truncate *elem_count*

Specifies the number of objects to print. The rest of the elements are truncated.

-no_sort

Does not sort the output data.

-objects *object_list*

Specifies the objects to format as a Tcl list.

This option is mutually exclusive with the *obj_spec_list* argument; you must specify either this option or the *obj_spec_list* argument.

obj_spec_list

Specifies the objects to format as a Tcl list.

This option is mutually exclusive with the **-objects** option; you must specify either this argument or the **-objects** option.

DESCRIPTION

The **collection_to_list** command prints a formatted Tcl list that corresponds to the collection provided as input. The Tcl list is sorted based on the object type and name (or only on the object name if you use the **-name_only** option). To output the list without sorting, use the **-no_sort** option.

EXAMPLES

The following example prints the Tcl list that corresponds to the collection passed as input.

```
prompt> collection_to_list [get_pins]
{{pin FF0/CLK} {pin FF0/D} {pin FF0/Q} {pin FF1/CLK} {pin FF1/D}
{pin FF1/Q} {pin SUB/A[0]} {pin SUB/A[1]} {pin U1/A} {pin U1/Y}}
```

The following example prints the object name only (without the object type) as a Tcl list.

```
prompt> collection_to_list [get_pins] -name_only
{FF0/CLK FF0/D FF0/Q FF1/CLK FF1/D FF1/Q {SUB/A[0]} {SUB/A[1]} U1/A U1/Y}
```

The follow example prints the Tcl list without beginning and ending curly braces.

```
prompt> collection_to_list [get_pins] -name_only -no_braces
FF0/CLK FF0/D FF0/Q FF1/CLK FF1/D FF1/Q {SUB/A[0]} {SUB/A[1]} U1/A U1/Y
```

The follow example prints each object on a new line.

```
prompt> collection_to_list [get_pins] -newline
{{pin FF0/CLK}
{pin FF0/D}
{pin FF0/Q}
{pin FF1/CLK}
{pin FF1/D}
{pin FF1/Q}
{pin SUB/A[0]}
{pin SUB/A[1]}
{pin U1/A}
{pin U1/Y}}
```

The follow example prints the first five objects with each object on a new line.

```
prompt> collection_to_list [get_pins] -newline -truncate 5
{{pin FF0/CLK}
{pin FF0/D}
{pin FF0/Q}
{pin FF1/CLK}
{pin FF1/D}}
```

collections

Describes the methodology for creating collections of objects and querying objects in the database.

DESCRIPTION

Synopsys applications build an internal database of objects and attributes applied to them. These databases consist of several classes of objects, including designs, libraries, ports, cells, nets, pins, clocks, and so on. Most commands operate on these objects.

By definition:

A collection is a group of objects exported to the Tcl user interface.

Collections have an internal representation (the objects) and, sometimes, a string representation. The string representation is generally used only for error messages.

Collections represent an ordered sequence of database objects. These collections provide constant time random access to the objects contained in the sequence.

A set of commands to create and manipulate collections is provided as an integral part of the user interface. The collection commands encompass two categories: those that create collections of objects for use by another command, and one that queries objects for viewing. The result of a command that creates a collection is a Tcl object that can be passed along to another command. For a query command, although the visible output looks like a list of objects (a list of object names is displayed), the result is an empty string.

An empty string "" is equivalent to the empty collection, that is, a collection with zero elements.

To illustrate the usage of the common collection commands, the man pages have examples. Most of the examples use PrimeTime as the application. In all cases, the application from which the example is derived is indicated.

Homogeneous and Heterogeneous Collections

A homogeneous collection contains only one type of object. A heterogeneous collection can contain more than one type of object. Commands that accept collections as arguments can accept either type of collection.

Lifetime of a Collection

Collections are active only as long as they are referenced. Typically, a collection is referenced when a variable is set to the result of a command that creates it or when it is passed as an argument to a command or a procedure. For example, in PrimeTime, you can save a collection of design ports by setting a variable to the result of the **get_ports** command:

```
pt_shell> set ports [get_ports *]
```

Next, either of the following two commands deletes the collection referenced by the *ports* variable:

```
pt_shell> unset ports
pt_shell> set ports "value"
```

Collections can be implicitly deleted when they go out of scope. Collections go out of scope for various reasons. An example would be when the parent (or other antecedent) of the objects within the collection is deleted. For example, if our collection of ports is owned by a design, it is implicitly deleted when the design that owns the ports is deleted. When a collection is implicitly

deleted, the variable that referenced the collection still holds a string representation of the collection. However, this value is useless because the collection is gone, as illustrated in the following PrimeTime example:

```
pt_shell> current_design
{"TOP"}

pt_shell> set ports [get_ports in*]
{"in0", "in1"}

pt_shell> remove_design TOP
Removing design 'TOP'...

pt_shell> query_objects $ports
Error: No such collection '_sel26' (SEL-001)
```

Iteration

To iterate over the objects in a collection, use the **foreach_in_collection** command. You cannot use the Tcl-supplied **foreach** iterator to iterate over the objects in a collection, because the **foreach** command requires a list, and a collection is not a list. In fact, if you use the **foreach** command on a collection, it destroys the collection.

The arguments of the **foreach_in_collection** command are similar to those of foreach: an iterator variable, the collection over which to iterate, and the script to apply at each iteration. Note that unlike the **foreach** command, the **foreach_in_collection** command does not accept a list of iterator variables.

The following example is an iterative way to perform a query in PrimeTime. For more information, see the **foreach_in_collection** man page.

```
pt_shell> \
foreach_in_collection s1 $collection {
  echo [get_object_name $s1]
  }
```

Manipulating Collections

A variety of commands are provided to manipulate collections. In some cases, a particular command might not operate on a collection of a specific type. This is application-specific. Consult the man pages from your application.

- **add_to_collection** - This command creates a new collection by adding a list of element names or collections to a base collection. The base collection can be the empty collection. The result is a new collection. In addition, the **add_to_collection** command allows you to remove duplicate objects from the collection by using the *-unique* option.
- **append_to_collection** - This command appends a set of objects (specified by name or collection) to an existing collection. The base collection is passed in through a variable name, and the base collection is modified directly. It is similar in function to the **add_to_collection** command, except that it modifies the collection in place; therefore, it is much faster than the **add_to_collection** command when appending.
- **remove_from_collection** - This command removes a list of element names or collections from an existing collection. The second argument is the specification of the objects to remove and the first argument is the collection to have them removed from. The result of the command is a new collection. For example, in PrimeTime:

  ```
  pt_shell> set dports \
  [remove_from_collection [all_inputs] CLK]
  {"in1", "in2", "in3"}
  ```

- **compare_collections** - This command verifies that two collections contain the same objects (optionally, in the same order). The result is "0" on success.
- **copy_collection** - This command creates a new collection containing the same objects in the same order as a given

collection. Not all collections can be copied.

- **index_collection** - This command extracts a single object from a collection and creates a new collection containing that object. The index operation is done in constant time - it is independent of the number of elements in the collection, or the specific index. Not all collections can be indexed.
- **sizeof_collection** - This command returns the number of objects in a collection.

Filtering

In some applications, you can filter any collection by using the **filter_collection** command. This command takes a base collection and creates a new collection that includes only those objects that match an expression.

Some applications provide a *-filter* option for their commands that create collections. This allows objects to be filtered out before they are ever included in the collection. Frequently this is more efficient than filtering after the they are included in the collection. The following examples from PrimeTime filters out all leaf cells:

```
pt_shell> filter_collection \
[get_cells *] "is_hierarchical == true"]
{"i1", "i2"}
pt_shell> get_cells * -filter "is_hierarchical == true"
{"i1", "i2"}
```

The basic form of a filter expression is a series of relations joined together with AND and OR operators. Parentheses are also supported. The basic relation contrasts an attribute name with a value through a relational operator. In the previous example, *is_hierarchical* is the attribute, *==* is the relational operator, and *true* is the value.

The relational operators are

```
==   Equal
!=   Not equal
>    Greater than
<    Less than
>=   Greater than or equal to
<=   Less than or equal to
=~   Matches pattern
!~   Does not match pattern
```

The basic relational rules are

- String attributes can be compared with any operator.
- Numeric attributes cannot be compared with pattern match operators.
- Boolean attributes can be compared only with == and !=. The value can be only true or false.

Additionally, existence relations determine if an attribute is defined or not defined, for the object. For example,

```
(sense == setup_clk_rise) and defined(sdf_cond)
```

The existence operators are

```
defined
undefined
```

These operators apply to any attribute as long as it is valid for the object class. See the appropriate man pages for complete details.

Sorting Collections

In some applications, you can sort a collection by using the **sort_collection** command. It takes a base collection and a list of attributes as sort keys. The result is a copy of the base collection sorted by the given keys. Sorting is ascending, by default, or descending when you specify the *-descending* option. In the following example from PrimeTime, the command sorts the ports by direction and then by full name.

```
pt_shell> sort_collection [get_ports *] \
{direction full_name}
{"in1", "in2", "out1", "out2"}
```

Implicit Query of Collections

In many applications, commands that create collections implicitly query the collection when the command is used at the command line. Consider the following examples from PrimeTime:

```
pt_shell> set_input_delay 3.0 [get_ports in*]
1
pt_shell> get_ports in*
{"in0", "in1", "in2"}
pt_shell> query_objects -verbose [get_ports in*]
{"port:in0", "port:in1", "port:in2"}
pt_shell> set iports [get_ports in*]
{"in0", "in1", "in2"}
```

In the first example, the **get_ports** command creates a collection of ports that is passed to the **set_input_delay** command. This collection is not the result of the primary command (**set_input_delay**), and as soon as the primary command completes, the collection is destroyed. The second example shows how a command that creates a collection automatically queries the collection when that command is used as a primary command. The third example shows the verbose feature of the **query_objects** command, which is not available with an implicit query. Finally, the fourth example sets the variable iports to the result of the **get_ports** command. Only in the final example does the collection persist to future commands until iports is overwritten, unset, or goes out of scope.

SEE ALSO

add_to_collection(2)
as_collection(2)
append_to_collection(2)
compare_collections(2)
copy_collection(2)
filter_collection(2)
foreach_in_collection(2)
index_collection(2)
query_objects(2)
remove_from_collection(2)
sizeof_collection(2)
sort_collection(2)

color_macro_pins

Colors the pins of the macro block using ICV with colored runset.

SYNTAX

status **color_macro_pins**

DESCRIPTION

The **color_macro_pins** command assigns the colors to pins of instantiated macro blocks in hierarchical design having double pattern process layers. The color to a pin is assigned on the basis of track colors, though there may not be 100% alignment possible.

This command invokes the IC Validator tool with a colored runset either on the NDM lib or GDS to assign the colors to pins based on track colors.

This command supports the following inputs - GDS and NDM Lib. There are different app options required to be set based on the given input. In case of GDS input flow, parameters such as technology file, layer map file, track offset, preferred routing direction, rule type, track color preference and others can be specified through app options.

All the app options for this command are specifed under the category of signoff.color_macro.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **color_macro_pins** command on the current block using the TSMC 16nm node colored runset:

```
prompt> set_app_options -name signoff.color_macro.rule_type \
  -value "tsmc16"
prompt> color_macro_pins
```

The following example shows the color attribute, **mask_constraint**, set on the pin shapes:

```
prompt> get_attribute [get_shapes -of_objects [get_port <port-name>]] \
  mask_constraint
```

SEE ALSO

signoff_create_metal_fill(2)
signoff_check_drc(2)

commit_blackbox_timing

Commits black box timing information specified by **create_blackbox_constraint**, **create_blackbox_delay**, and other black box timing commands.

SYNTAX

status **commit_blackbox_timing**
[-convert_placement_abstract true | false]

ARGUMENTS

-convert_placement_abstract true | false

Converts the placement abstract to a timing abstract with black box timing (BBT) data. Specify **-convert_placement_abstract false** to prevent the command from overwriting an existing placement abstract view. By default, this option is true and the command overwrites the existing abstract view of the design and replaces it with a compact timing abstract view with black box timing.

DESCRIPTION

This command saves the black box timing information you created with the **create_blackbox_constraint**, **create_blackbox_delay**, and other black box timing commands. The command creates or updates the abstract view of the current block to include the black box timing data.

EXAMPLES

The following example creates an abstract view, with black box timing information, for the current design.

```
prompt> create_clock -period 10 -name main_clock [get_ports Clk]
prompt> set_blackbox_clock_port [get_ports Clk]
prompt> create_blackbox_clock_network_delay -value 0.5 [get_ports Clk]
prompt> create_blackbox_load_type -lib_cell buf0 loadType1
prompt> create_blackbox_drive_type -lib_cell buf8 driveType1
prompt> set_blackbox_port_load -type loadType1 [all_inputs]
prompt> set_blackbox_port_drive -type driveType1 [all_outputs]
prompt> create_blackbox_delay -rise_from Clk -to [all_outputs] -clock main_clock -value 0.3
prompt> create_blackbox_constraint -from Clk -edge rise -to [all_inputs] \
```

```
        -clock main_clock -value 0.3
prompt> commit_blackbox_timing
```

SEE ALSO

create_blackbox_clock_network_delay(2)
create_blackbox_constraint(2)
create_blackbox_delay(2)
create_blackbox_drive_type(2)
create_blackbox_load_type(2)
remove_blackbox_timing(2)
set_blackbox_clock_port(2)
set_blackbox_port_drive(2)
set_blackbox_port_load(2)
write_blackbox_timing_script(2)

commit_block

Creates a separate physical hierarchy design from a logical hierarchical cell.

SYNTAX

collection **commit_block**
[-library *library*]
[-verbose]
modules

Data Types

library string
modules collection

ARGUMENTS

-library *library*

Specifies the library in which to create the new design. By default, the current design library is used.

-verbose

Reports additional debugging information.

modules

Specifies the name of a logic cell to commit into a physical block. You can specify a module name or a collection containing one module object. This option is required.

DESCRIPTION

This command commits a logical hierarchy cell to a physical hierarchy block and returns a collection containing the committed cell.

The shape of the committed block is dependent upon whether cell hierarchy type is "module" or "module boundary". If it is "module", there is no physical boundary by which to determine a physical boundary of a resulting block. In this case, the command will create a default square boundary based upon criteria such as the number of leaf-cells/blocks in the module. Also in this case, the **commit_block** command will not attempt to place member cells within the committed reference block since it has no certain origin by which to place child cells inside the reference block. So in this scenario, the cells in the reference block will be unplaced. Also in this scenario, the instance of the block will be placed outside the top block core boundary.

If the module cell hierarchy type is "module boundary" and you have placed the module boundary in the core area, then the

command has what it needs to create a block boundary, and an origin that it can use to place the cells in the reference block, and will do so. In this scenario, the resulting block instance will be placed based upon the determined origin of the module boundary associated with the module instance. In any case, regardless of where the module boundary is placed, the resulting placement of cells in the committed block will show the exact same geometric relationship to the block origin in the reference block as the member cell instances of the module instance has to the lower left corner of the module boundary before **commit_block**. This is because **commit_blocks** merely transforms the relative locations from the top to the block based upon the cell instance's location relative to the module boundary lower-left corner location.

The **commit_block** command propagates routing, move bounds, relative placement groups and other logical-based elements to the block level during the commit process. All routing associated with module-owned nets is propagated automatically to the block. All move bounds that are enclosed by the module boundary, and whose cells are only those that belong to the module being committed, are propagated to the block. All group bounds whose cell members exclusively belong to the module instance are also propagated automatically to the block. All relative placement groups that are enclosed by the module boundary, and whose cells exclusively belonging to the module being committed, are propagated to the block.

Also as part of the commit process, certain overlap-based propagations occur. This includes placement blockages and route guides that are completely within the module boundary of the module being committed. If there is no module boundary, then placement blockages and route guides are not propagated.

If the module type is "module" (or normal), and the module instance has multiple fixed placed instances, the command auto-derives a bounding box defined by these instances, and creates a block boundary. The boundary is based on the bounding box and the block origin is placed at the lower-left corner of the bounding box, with R0 orientation. If there are multiple instances of this module with fixed placed instances, the command chooses a reference among the fixed-placed child instance of the module instance and creates and places the first block instance as mentioned above.

For each subsequent instance, the fixed-placed reference instance is used to determine the location and orientation the remaining MIB block instances. You must carefully confirm that ALL fixed-placed child instances are exactly consistent with each of the other MIB module instances' child instances. If the instances are not consistent, the results of committing the MIB instances are very unpredictable. The command DOES NOT pre-check that all MIB module instances are consistently placed. You must check the placement consistency before using the **commit_block** command to auto-infer shapes and orientations of MIB committed block instances.

If an instance of the committed block results in placement over any site arrays, copies of these site arrays are created within the block based upon that overlap with top-level site arrays. If there is no overlap (type is "module") with any site array after commit, then a default site array is still created in the block, based upon the top default site array, but this site array is not guaranteed to be aligned with the top site array because the committed block instance does not overlap the top default site array. This is generally not an issue because in this situation, it is normally the case that subsequent flow steps will involve block shaping, which will propagate the default site array after the block has been placed and shaped, and at that point the block-level site array should then be in alignment.

By default, a hard inner keepout margin is created in the committed block. The committed block's default inner keepout is equal to one row height, and its default outer keepout is equal to twice of the minimal wire track width.

Use the following application options to specify how the inner keepout is created:

plan.commit.inner_keepout_margin hard | soft | none

user_unit_distance must include unit # appended to the distance value plan.commit.inner_keepout_margin_size *user_unit_distance*

plan.commit.outer_keepout_margin hard | soft | none

user_unit_distance must include unit # appended to the distance value plan.commit.outer_keepout_margin_size *user_unit_distance*

examples:

set_app_options -list {plan.commit.inner_keepout_margin hard}

set_app_options -list {plan.commit.inner_keepout_margin_size 0.3um}

set_app_options -list {plan.commit.outer_keepout_margin hard}

set_app_options -list {plan.commit.outer_keepout_margin_size 0.1um}

set_app_options -list {plan.commit.inner_keepout_margin none}

set_app_options -list {plan.commit.outer_keepout_margin none}

Note that application option values are persistent in the database.

Note that the **commit_block** command fails under the following conditions:

- The specified library is not the design library or one of the current reference libraries.
- The module does not exist or is not a hierarchical logic cell.
- The library already contains a design with the name of the new design to be created.

EXAMPLES

The following example commits the cells that are instantiated from module A to blocks.

```
prompt> commit_block A
```

SEE ALSO

get_cells(2)
get_view_switch_list(2)
uncommit_block(2)

commit_secondary_pg_placement_constraints

commit all secondary pg constraints

SYNTAX

status **commit_secondary_pg_placement_constraints**

ARGUMENTS

There is no argument for this command

DESCRIPTION

After all secondary pg constraints are created, users need to issue this command so that the tool can process all the constraints and internally generate the physical regions which will allow dual rail cells to be inserted in.

If the floorplan, UPF, secondary supply straps or secondary pg constraints are changed or removed, users also need to issue this command so that the tool can process all the remain constraints and internally regenerate or remove any special physical regions created due to the prior existed secondary pg constraints.

EXAMPLES

The following example commit all secondary pg constraint

prompt> **commit_secondary_pg_placement_constraints**

SEE ALSO

create_secondary_pg_placement_constraints(2)
remove_secondary_pg_placement_constraints(2)
report_secondary_pg_placement_constraints(2)

commit_upf

Commit the UPF constraints of the design.

SYNTAX

status **commit_upf**
[-skip_resolve_pg_net]

ARGUMENTS

-skip_resolve_pg_net

If specified, the command will not resolve potential conflicts between power intent and pg net connections.

DESCRIPTION

This command commits the UPF intent of the design. It indicates that the UPF constraints of the design are finalized and complete.

By default, the command tries to resolve potential conflicts between the power intent and PG net connections. To disable this behavior, set the **mv.upf.auto_resolve_pg_net** application option to **false**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example commits the UPF intent of the design.

prompt> **commit_upf**

SEE ALSO

load_upf(2)

compare_app_options

Reads and compares application option settings between two tool sessions. Settings data is written to a binary data file by the **write_app_options** command.

SYNTAX

integer **compare_app_options**
-previous_data *data_file*
-current_data *data_file*
[-value_diffs]
[-csv *file_name*]
[-metadata]

Data Types

data_file string
file_name String. Report contents to this file in CSV format.

ARGUMENTS

-previous_data *data_file*

Specifies the file that contains the binary application option settings data. The *data_file* argument can be a local file name or a full path name to the file. The data specified by this option is the previous or original data and is displayed on the left side of the comparison output report.

-current_data *data_file*

Specifies the file that contains the binary application option settings data. The *data_file* argument can be a local file name or a full path name to the file. The data specified by this option is the current or data to be compared against the original and is displayed on the right side of the comparison output report.

-value_diffs

Generate an alternate report which combines all app option value differences into one table.

-csv *file_name*

Specifies the output CSV file name. The report contents will be output to this file in CSV format. The *file_name* argument can be a local file name or a full path name to the file.

-metadata

Add the metadata which includes comment information of compared data to the CSV file.

DESCRIPTION

This command creates a single report with multiple tables reporting these differences for each application option that differs between the two input files:

- Status
- Changes in Default Value
- Changes in Global Value
- Changes in Block Value
- Changes in Global status
- Changes in ReadOnly status
- Changes in Deprecated status
- Changes in Persistent status
- Changes in Disabled status
- Changes in Application Variables

The report will also show Added and Removed application options.

- Added application options are in the current data file, but not in the previous_data file
- Removed application options are in the previous data file, but not in the current_data file

The command writes the report to the console. To save the output to a .csv file, you could run the command as follows:

```
compare_app_options \
   -previous_data file1.data \
   -current_data file2.data \
   -csv compare.csv
```

Use the **compare_app_options** command to generate reports such as:

- Application option differences when block A is loaded vs. block B
- Application option differences between different startup Tcl scripts
- Application option differences between different tool releases

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example generates the binary application option data files for two tool sessions and compares the files.

```
# In the first tool session
prompt> write_app_options -output app_option_SP1.data
```

```
# In the second tool session
prompt> write_app_options -output app_option_SP2.data
prompt> compare_app_options -previous_data app_option_SP1.data \
  -current_data app_option_SP2.data
```

This example writes the binary application option data and shows the differences in application options between design A and design B.

```
# Design A is now the current design
prompt> current_design designA
prompt> write_app_options -output designA.data

# Design B is now the current design
prompt> current_design designB
prompt> write_app_options -output designB.data

# Compare application options
prompt> compare_app_options -previous_data designA.data \
  -current_data designB.data
```

The following example shows a sample report generated by the **compare_app_options** command.

```
prompt> compare_app_options -previous_data old.data \
  -current_data new.data

Opened Previous File: old.data
    Total: 1176
    ------ -----
    basic: 1176

Opened Current File: new.data
    Total: 1181
    ------ -----
    basic: 1181

Summary:
-------------------------

Added App Options:
    Total: 10
    ------ -----
    basic: 10

Removed App Options:
    Total: 5
    ------ -----
    basic: 5

Difference Summary:
    Default Value   4
    Block Value     1
    Global          1
    Persistent      1

Differences:
-------------------------
```

```
Changes in: Status                          Previous     Current
------------------------------------------------------ ------------ ------------

Changes in: Default Value                     Status     Previous     Current
------------------------------------------------------ ---------- ------------ ------------
place.coarse.balance_registers                  basic      false        true
place_opt.initial_drc.global_route_based        basic      0            false
rail.allow_redhawk_license_checkout             basic      false        on_demand
shell.common.product_build_date                 basic      Oct 01, 2018 Nov 01, 2018

Changes in: Global Value                      Status     Previous     Current
------------------------------------------------------ ---------- ------------ ------------

Changes in: Block Value                       Status     Previous     Current
------------------------------------------------------ ---------- ------------ ------------
power.scale_leakage_power_at_power_off          basic                true

Changes in: Global                          Status     Previous     Current
------------------------------------------------------ ---------- ------------ ------------
mv.upf.write_crosstool_wrappers                 basic      global

Changes in: ReadOnly                        Status     Previous     Current
------------------------------------------------------ ---------- ------------ ------------

Changes in: Deprecated                      Status     Previous     Current
------------------------------------------------------ ---------- ------------ ------------

Changes in: Persistent                      Status     Previous     Current
------------------------------------------------------ ---------- ------------ ------------
mv.upf.write_crosstool_wrappers                 basic      false        true

Changes in: Disabled                        Status     Previous     Current
------------------------------------------------------ ---------- ------------ ------------

Added
-------------------------
ccd.fmax_optimization_effort                  basic
cts.compile.power_opt_mode                    basic
da.check_netlist.allow_connection_class_violation    basic      Global
da.check_netlist.check_for_wire_loop          basic      Global
lib.configuration.local_output_dir            basic      Global
lib.configuration.remove_local_output_dir     basic      Global
opt.common.advanced_power_restructuring_step  basic
place.coarse.cong_restruct_depth_aware        basic
place.legalize.enable_cross_row_pnet_check    basic

Removed:
-------------------------
lib.configuration.output_dir                  basic
lib.configuration.remove_output_dir           basic
place.coarse.cong_restruct_depth_limit        basic
rail.enable_all_redhawk_report_files          basic
route_opt.flow.enable_metal_fill_within_ropt  basic
```

The following example compares two data files and output the result to the specified file as CSV format using the '-csv' option.

```
prompt> compare_app_options -previous_data old.data \
  -current_data new.data -csv compare.csv

compare.csv
```

SEE ALSO

get_app_options(2)
help_app_options(2)
report_app_options(2)
reset_app_options(2)
set_app_options(2)
write_app_options(2)

compare_checksum

This utility is for comparing checksums generated by write_checksum.

SYNTAX

status **compare_checksum**
-icc2 *icc2_checksum_path*
-pt *pt_checksum_path*

Data Types

icc2_checksum_path string
pt_checksum_path string

ARGUMENTS

-icc2 ***icc2_checksum_path***

Specifies the directory that contains the IC Compiler II checksums to compare. This option is required.

-pt ***pt_checksum_path***

Specifies the directory that contains the PrimeTime checksums to compare. This option is required.

DESCRIPTION

The **compare_checksum** compares checksums generated by write_checksum of input consistency checker.

The command would run in the background. The output of the command is redirected to

- correlation.out

  The compare checksum reports files will be

  - DesignChecksumSummary.rpt

  - DesignChecksumDetail.rpt
  - LibraryChecksumSummary.rpt
  - LibraryChecksumDetail.rpt
  - ParasiticsChecksumSummary.rpt
  - ParasiticsChecksumDetail.rpt
  - ConstraintsCheckerSummary.rpt

- ConstraintsCheckerDetail.rpt

By default, the **write_checksum** command generates all the checksums in the current directory.

EXAMPLES

The following example generates compare checksum reports in the current directory. The IC Compiler II checksums are in ./icc2/scenario and the PrimeTime checksums are in ./pt.

```
prompt> compare_checksum -icc2 ./icc2/scenario1 -pt ./pt
```

SEE ALSO

write_checksum(2)

compare_collections

Compares the contents of two collections. If the same objects are in both collections, the result is "0" (like string compare). If they are different, the result is nonzero. The order of the objects can optionally be considered.

SYNTAX

int **compare_collections**
[-order_dependent]
collection1
collection2

Data Types

collection1	collection
collection2	collection

ARGUMENTS

-order_dependent

Indicates that the order of the objects is to be considered; that is, the collections are considered to be different if the objects are ordered differently.

collection1

Specifies the base collection for the comparison. The empty string (the empty collection) is a legal value for the *collection1* argument.

collection2

Specifies the collection with which to compare to *collection1*. The empty string (the empty collection) is a legal value for the *collection2* argument.

DESCRIPTION

The **compare_collections** command is used to compare the contents of two collections. By default, the order of the objects does not matter, so that a collection of cells u1 and u2 is the same as a collection of the cells u2 and u1. By using the **-order_dependent** option, the order of the objects is considered.

Either or both of the collections can be the empty string (the empty collection). If two empty collections are compared, the comparison succeeds (that is, **compare_collections** considers them identical), and the result is "0".

EXAMPLES

The following example from PrimeTime shows a variety of comparisons. Note that a result of "0" from **compare_collections** indicates success. Any other result indicates failure.

```
pt_shell> compare_collections [get_cells *] [get_cells *]
0
pt_shell> set c1 [get_cells {u1 u2}]
{"u1", "u2"}
pt_shell> set c2 [get_cells {u2 u1}]
{"u2", "u1"}
pt_shell> set c3 [get_cells {u2 u4 u6}]
{"u2", "u4", "u6"}
pt_shell> compare_collections $c1 $c2
0
pt_shell> compare_collections $c1 $c2 -order_dependent
-1
pt_shell> compare_collections $c1 $c3
-1
```

The following example builds on the previous example by showing how empty collections are compared.

```
pt_shell> set c4 ""
pt_shell> compare_collections $c1 $c4
-1
pt_shell> compare_collections $c4 $c4
0
```

SEE ALSO

collections(2)

compare_floorplans

Writes a report that lists any differences in objects and their locations between the current design and floorplan information for another design written by the **write_floorplan** command.

SYNTAX

int **compare_floorplans**
[-input *dir_name*]
[-verbosity minimum | low | high]
[-top_level_only]

Data Types

dir_name string

ARGUMENTS

-input *dir_name*

Specifies the name of the directory under which the file *floorplan_compare_data.txt* is located. The default is *./floorplan*. The command writes an error message and exits if the directory does not exist, or if the file is missing.

-verbosity minimum | low | high

Determines the level of comparison detail to report. The **minimum** argument prints the number of differences for each object type. The **low** argument prints up to five examples for each object type. The **high** argument prints up to 1000 differences of each type. By default, the level is minimum.

-top_level_only

Compares only top-level objects and skip comparison for lower-level objects.

DESCRIPTION

The **compare_floorplans** command creates a report containing a list of objects that are either in *floorplan_compare_data.txt* file but not in the design database, or in the design database but not in *floorplan_compare_data.txt*. The object types that are considered are macros and pins. Feedthrough pins are not considered. The command creates a file named *compare_output*, in the current directory, that creates lists of new objects that are in the existing in database but not in the file. For example:

```
set _new_macros [get_cells { macro1_name macro2_name ...}]
set _new_ports {...}
```

For objects in the *floorplan_compare_data.txt* file but not in the database, the command outputs names and locations or boundaries in the same file. The command outputs two levels, for example, pins and macros at the top level and inside blocks. The command skips the output for the third level and beyond.

Note that the *floorplan_compare_data.txt* file is generated by the **write_floorplan** command. The **compare_floorplans** command is typical used to compare different versions of the same design. For example, begin by running the **write_floorplan** command on one version of the design. Then, load the new version and use the **compare_floorplans** command to determine if any differences exist between the two versions.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following example writes a report based on comparing the current design with the *floorplan_compare_data.txt* file in the script directory.

```
prompt> compare_floorplans -input script
```

The following example prints more detailed information, but outputs data related to the top-level only.

```
prompt> compare_floorplans -verbosity high -top_level_only
```

SEE ALSO

write_floorplan(2)

compare_supplies

compares voltage levels or relative ON-ness between two given supplies.

SYNTAX

status **compare_supplies**
supply_list
[-on_status]
[-voltage_level]

Data Types

supply_list list
on_status boolean
voltage_level boolean

ARGUMENTS

supply_list

Specifies a list of 2 supplies. The first supply specified is treated as the 'reference' supply to which the second supply ('specified' supply) will be compared to. Supplies can either be supply net names or supply set handles.

-on_status

If specified, this option will return one of 4 values: "equal" if the reference supply is equal to the specified supply. "more_on" if the specified supply is more ON as compared to the reference supply. "less_on" if the specified supply is less ON as compared to the reference supply. "independent" if the specified supply may be more or less ON as compared to the reference supply depending on the PST State.

-voltage_level

If specified, this option will return one of 4 values: "equal", if both reference and specified supplies are at the same voltage and no level-shifting is needed. "higher", if reference supply is lower voltage as compared to the specified supply. "lower", if the reference supply is higher voltage as compared to the specified supply. "independent, if the reference supply is higher/lower voltage as compared to the specified supply depending on the PST State. If the two supplies specified belong to the same correlated supply group, they will be compared based on the rules for supplies in the same correlated supply group.

DESCRIPTION

This command is used to compare voltage levels or relative always-ON levels of two given supplies. Options -on_status and -

voltage_level can be used separately or in combination. Supplies list must consist of exactly 2 supplies. If the specified supply has no states defined, we will treat it as equal to the reference supply and return equal for both on_status and voltage_level.

SEE ALSO

report_pst(2)

compile

Runs synthesis flow

SYNTAX

status **compile**
[-no_boundary_optimization]
[-no_autoungroup]
[-checkpoint]

Data Types

ARGUMENTS

-no_boundary_optimization

Disables optimization across hierarchical boundaries.

-no_autoungroup

Disables automatic ungrouping, preserving hierarchy.

-checkpoint

Writes 2 intermediate checkpoint databases.

DESCRIPTION

This command runs the synthesis flow. Note: This command is deprecated. It is replaced with **compile_fusion** command.

EXAMPLES

prompt> **compile**

SEE ALSO

compile_fusion(2)

compile_boundary_cells

Creates and places boundary cells into a design. This command is used to replace the command *create_boundary_cells*.

SYNTAX

status **compile_boundary_cells**
  [-add_placement_blockage]
  [-voltage_area {*objects*}]

Data Types

objects list

ARGUMENTS

-add_placement_blockage

Add placement blockage to meet boundary cells rules.

-voltage_area {*objects*}

Specifies the voltage areas to insert boundary cells.

DESCRIPTION

This command adds boundary cells around the boundaries of objects, such as voltage areas, macros, blockages, and the core area, according to the boundary cell rules set by *set_boundary_cell_rules*.

EXAMPLES

The following example creates boundary cells on both the left and the right edges of boundaries.

```
prompt> set_boundary_cell_rules -left_boundary_cell myLib/CellLeft \
        -right_boundary_cell myLib/CellRight
prompt> compile_boundary_cells
```

SEE ALSO

check_boundary_cells(2)
check_targeted_boundary_cells(2)
compile_targeted_boundary_cells(2)
remove_boundary_cell_rules(2)
report_boundary_cell_rules(2)
set_boundary_cell_rules(2)

compile_fusion

Runs the flow of synthesis, placement, and optimization.

SYNTAX

compile_fusion
[-list_only]
[-from *startStage*]
[-to *endStage*]
[-check_only]

Data Types

startStage string
endStage string

ARGUMENTS

-list_only

Shows the stages that constitute the **compile_fusion** command.

This option cannot be used with the **-from** or **-to** option.

-from *startStage*

Specifies the starting stage for the **compile_fusion** command. Valid values are any of the seven stages reported by *-list_option* option.

The default value is *initial_map* stage similar to classic **compile** command

-to *endStage*

Specifies the final stage for the **compile_fusion** command. Valid values are any of the seven stages.

The default value is **final_opto** with legalized placement similar to **place_opt** command.

If both the **-to** and **-from** options are specified, the *startStage* stage cannot be preceded by the *endStage* stage in the order specified by the **-list_only** option.

-check_only

Checks any discrepancy found in the library and design.

If any of the **-to** or **-from** option is specified, the library and design will be checked to be sufficient only for the stages that will actually run.

DESCRIPTION

By default, the command performs the following tasks in sequence:

- Maps the generic cells to library cells and performs logic optimization.
- Optimizes timing based on logic only, not considering physical data.
- Places the current design.
- Optimizes the placed design for timing, electrical DRC violations, area, power, and routability.
- Performs incremental placement to optimize timing and routability.
- Further optimizes the placed design.
- legalizes the design placement at the end.

EXAMPLES

The following example run all seven stages of the flow. It starts from logic synthesis and ends up with an optimized design with legalized placement:

```
prompt> compile_fusion
```

The following example runs only the **initial_opto** stage. It performs incremental placement to optimize timing and routability:

```
prompt> compile_fusion -from initial_opto -to initial_opto
```

The following example runs the last stages of the flow. It takes in a placed and optimized design as its input, further optimizes the placed design, and ends up with an optimized design with legalized placement:

```
prompt> compile_fusion -from final_place
```

The following example runs the post-compile DFT insertion flow It takes in a placed and optimized design as its input, performs DFT insertion and ends up with an optimized design with legalized placement:

```
prompt> compile_fusion -to initial_opto
prompt> insert_dft
prompt> compile_fusion -from final_place
```

The following example migrates some pre-place_opt settings to the middle of compile_fusion for a unified UPS flow.

```
prompt> compile_fusion -to initial_opto
prompt> source physical_opto_settings.tcl
prompt> compile_fusion -from final_place
```

SEE ALSO

place_opt(2)

compile_pg

Creates a power and ground network including straps, rings, macro connections, standard cell connections and via connections based on the specified power ground strategies.

SYNTAX

status **compile_pg**
[-strategies *strategy_list*]
[-via_rule *rule_list*]
[-ignore_drc]
[-ignore_via_drc]
[-show_phantom]
[-undo]
[-tag *tag_name*]
[-create_ml_data]
[-use_ml_model]

Data Types

strategy_list list
rule_list list
tag_name string

ARGUMENTS

-strategies *strategy_list*

Creates a power network based only on the specified list of strategies. By default, the tool creates a power ground network using all strategies which have been defined.

-via_rule *rule_list*

Specifies the via rule list between strategies and existing shapes. The via rule list *rule_list* contains strategy-level via rules defined by the **set_pg_strategy_via_rule** command. If no strategy via rule is specified, the default via defined in the technology file is created at all intersections between any two orthogonal shapes on different layers.

-ignore_drc

Creates the power network while ignoring DRCs for both wires and vias. By default, all DRCs are considered when creating the power network. This option is mutually exclusive with the **-ignore_via_drc** option.

-ignore_via_drc

Creates the power network while ignoring via DRCs. By default, all DRCs are considered when creating the power network. When this option is set, DRCs for wires are checked and fixed, DRCs within each stacked via are checked and fixed. Only DRCs

between stacked vias and neighboring objects are ignored. This option is mutually exclusive with **-ignore_drc** option.

-show_phantom

Generates report for wires/vias that should be created but failed due to fixing DRC violation. Detailed information will be displayed in error browser. The report includes two parts, phantom wire and phantom via. Wire information and DRC violation are included in one phantom wire. Only DRC violations that cut part of wire shape while fixing will be included in phantom wire. Abutted phantom wires will be merged and reported together. Via information and DRC violation are included in one phantom via. Via information includes top/bottom wire information, via def, net name. DRC violation shows initial violation before fixing. If **-ignore_drc** or **-ignore_via_drc** is specified, phantom via will not be generated.

-undo

Removes the power and ground network created by the most recent **compile_pg** command.

-tag *tag_name*

Specifies a tag name to assign to all new vias and shapes created by this command. This tag name is used as contents of an user attribute tag for filtering collections at later stages (see the example in the EXAMPLES section). By default, no tags are assigned.

-create_ml_data

This option enables this command to create training data for training ML model for fixability prediction. When this option is set, the command only creates ML training data, not creating PG wires/vias.

-use_ml_model

This option enables this command to load pre-trained ML model for fixability prediction.

DESCRIPTION

This command creates the power network based on the specified power ground strategies.

Signal routes by default are ignored for DRC checking and correction during PG creation. Use the following app option to honor signal route DRC during PG creation.

```
prompt> set_app_options -list {plan.pgroute.honor_signal_route_drc true}
```

EXAMPLES

The following example creates a power ground network using only the example1 and example2 strategies. The rule1 via rule is defined by the **set_pg_strategy_via_rule** command.

```
prompt> compile_pg -strategies {example1 example2}\
            -via_rule rule1
```

The following example compiles strategy example1 using 4 threads defined in the global host_options mt for via creation.

```
prompt> set_host_options -max_cores 4
prompt> compile_pg -strategies {example1}
```

The following example creates collections of vias and wires that are tagged with the name pg1.

```
prompt> compile_pg -tag pg1
prompt> set vias [get_vias -filter tag==pg1]
prompt> set wires [get_shapes -filter tag==pg1]
```

The following example creates ML training data in memory, output training data to files in the directory ./pg_model, and train ML model based on the training data. The trained ML model is stored in the directory ./pg_model.

```
prompt> compile_pg -create_ml_data
prompt> train_pg_ml_model
```

The following example creates ML training data in memory, output training data to files in directory pg_data_1, and train ML model based on the training data. The trained ML model is stored in the directory ./pg_model.

```
prompt> compile_pg -create_ml_data
prompt> create_pg_ml_data -output_directory pg_data_1
prompt> train_pg_ml_model -input_directory {pg_data_1}
```

The following example loads pre-trained ML model in the directory ./pg_model for fixability prediction to reduce runtime of PG via creation.

```
prompt> compile_pg -use_ml_model
```

In addition to the above examples, see the Examples page in the PG Planning section of the Task Assistant for more information. To view the Examples page, start the GUI and invoke the following command: **gui_show_task_assistant -task "Design Planning:PG Planning->Examples->Overview"**.

SEE ALSO

remove_pg_strategies(2)
report_pg_strategies(2)
report_pg_strategy_via_rules(2)
set_host_options(2)
set_pg_strategy(2)
set_pg_strategy_via_rule(2)
get_vias(2)
get_shapes(2)
create_pg_ml_data(2)
train_pg_ml_model(2)

compile_targeted_boundary_cells

Creates and places boundary cells around target objects in current design.

SYNTAX

status **compile_targeted_boundary_cells**
-target_objects {*target_objects*}
[-include_touching_placement_blockages]
[-ignore_row_orientation]

Data Types

target_objects list

ARGUMENTS

-target_objects {*target_objects*}

Specifies the target objects to be around with boundary cells in current design. Objects including macro, voltage area, voltage area shape, placement blockage, route blockage and core area can be specified. This option is a required option.

-include_touching_placement_blockages

Specifies that when boundary cells are created and placed, placement blockages those touch specified target objects should be included. By default, no touching placement blockages are included.

-ignore_row_orientation

Specifies that when flipping between top and bottom.

DESCRIPTION

This command adds boundary cells around the boundaries of target objects, such as voltage areas, macros, blockages, and the core area, according to the boundary cell rules set by *set_boundary_cell_rules*.

EXAMPLES

The following example creates boundary cells around target placement blockages on both the left and the right edges of

boundaries.

```
prompt> set_boundary_cell_rules -left_boundary_cell myLib/CellLeft \
        -right_boundary_cell myLib/CellRight
prompt> compile_targeted_boundary_cells -target_objects {PB_7 PB_8}
```

SEE ALSO

check_boundary_cells(2)
check_targeted_boundary_cells(2)
compile_boundary_cells(2)
remove_boundary_cell_rules(2)
report_boundary_cell_rules(2)
set_boundary_cell_rules(2)

compute_area

Calculates the area of a geometric region.

SYNTAX

float **compute_area**
[-objects *object_list*]
[*pos_object_list*]

Data Types

object_list collection
pos_object_list collection

ARGUMENTS

-objects *object_list*

Specifies the objects to be used to define the geometric region to calculate the area of. Objects may be a heterogenous collection of poly_rects, geo_masks, shapes, layers, and other physical objects.

In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

This is a required option if the *pos_object_list* option is not specified.

pos_object_list

Specifies the objects to be used to define the geometric region to calculate the area of. This positional option is provided for compatibility with other tools.

DESCRIPTION

This command calculates the area of the region specified by *objects*.

SEE ALSO

copy_to_layer(2)
create_poly_rect(2)
create_geo_mask(2)
compute_polygons(2)
resize_polygons(2)
split_polygons(2)
transform_polygons(2)

compute_budget_constraints

Creates budget pin constraints based on virtual in-place optimization (VIPO) timing.

SYNTAX

status **compute_budget_constraints**
[-setup_delay]
[-boundary]
[-latency_targets actual | estimated]
[-busplans *buses*]
[-modes *mode_list*]
[-estimate_timing]
[-no_estimate_timing]
[-pins *pin_list*]
[-input]
[-output]
[-unspecified]
[-feedthrough]
[-slack *slack_value*]
[-fanin_cone]
[-fanout_cone]
[-balance true | false]
[-ocv_percent *percent*]
[-ocv_delay *delay*]

Data Types

buses list
mode_list list
pin_list list
slack_value float
percent float
delay float

ARGUMENTS

-setup_delay

Calculates budget values for setup delay. See the description section for more details. This can be specified in combination with the **-boundary**, and/or **-latency_targets** options. If none are specified, setup and boundaries are computed.

-boundary

Calculates budget values for drives and loads at the block boundary. See the description section for more details. This can be

specified in combination with the **-setup_delay**, and/or **-latency_targets** options. If none are specified, setup and boundaries are computed.

-latency_targets actual | estimated

Calculates new clock latency targets to be used during budgeting. This can be specified in combination with the **-setup_delay**, and/or **-boundary** options. If none are specified, setup and boundaries are computed. No other budget values are changed. During budget generation, you can use the clock latency targets to account for clock tree variation early in the design flow before clock tree synthesis (CTS). This option automatically sets the budget parameters for clock latency targets in your design. This is equivalent to manually using the **-early_latency**, **-late_latency** and **-crp** options with the **set_latency_budget_constraints** command. The **-latency_targets** option must be specified together with the **-balance** option.

The argument to **-latency_targets** determines how the budget values are selected.

- **actual**: Budget values are selected to closely match the current clock latencies in your top-level design. The latencies can be either ideal or propagated, depending on whether your SDC file contains **set_propagated_clock** commands.
- **estimated**: Budget values are selected to closely match the predicted latency of your top-level clock tree. The prediction is based on the current placement of your clock tree elements (the clock trunk, ICGs, clock pins). The current timing of the clock tree is not used. Estimated clock latencies are queried from the "clock trunk planning" tool, so you need to make sure that you have run that first.

-busplans *buses*

Automatically determines a fixed budget for segments along a budget path. This option is equivalent to manually setting the **-delay** option of the **set_segment_budget_constraints** command. The "buses" argument passed to this option must first have been declared using the **create_budget_busplan** command. See the **create_budget_busplan** man page for more details.

-modes *mode_list*

Specifies the list of modes to use to create the budget constraints. If this option is not specified, the command processes all modes with an active corner.

-estimate_timing

The automatic budgeter can run in one of two modes. In regular mode, automatic budgeting creates constraints based on the worst delays across all active timing corners. In estimate_timing mode, the automatic budgeter calculates the budget by using the delays in the estimated_corner. This is desirable when some or all of your design is unoptimized. Normally the mode is decided by the **plan.budget.estimate_timing_mode** application option. The **-estimate_timing** option uses the estimate_timing mode for calculating constraints.

-no_estimate_timing

The automatic budgeter can run in one of two modes. In regular mode, automatic budgeting creates constraints based on the worst delays across all active timing corners. In estimate_timing mode, the automatic budgeter will base the budget only on delays in the estimated_corner. This is desirable when some or all of your design is unoptimized. Normally the mode is decided by the **plan.budget.estimate_timing_mode** application option. The **-no_estimate_timing** option uses the regular mode for calculating constraints.

-pins *pin_list*

Updates budget values for only the specified pins. You can further restrict the list of pins with the **-input** and **-output** options. By default, the command processes all pins of all budgeted blocks.

-input

Updates budget values for only input pins to blocks. You can further restrict the list of pins with the **-pins** option. By default, the command processes all pins of all budgeted blocks.

-output

Updates budget values for only output pins to blocks. You can further restrict the list of pins with the **-pins** option. By default, the command processes all pins of all budgeted blocks.

-unspecified

Updates budget values for only pins with no existing delay budget. By default, the command processes all pins of all budgeted blocks.

-feedthrough

Updates budget values only for paths with feedthroughs. A feedthrough path has a combinational connection that goes from the input of a block to the output of a block. By default, the command updates all paths, including those that start or end in the local block.

-slack *slack_value*

Updates budget values only for paths through pins where the budget slack is less than **slack_value**. Budget slack is defined as the amount by which the actual path delay is less than the budgeted path delay. If you specify **-slack 0.0**, only paths where the budget is tighter than the actual delay are rebudgeted.

-fanin_cone

Updates budget values only for pins that are in the timing fanin cone of the selected pins. This is useful when you want to rebudget many pins along the same path. By default, only selected pins are processed.

-fanout_cone

Updates budget values only for pins that are in the timing fanout cone of the selected pins. This is useful when you want to rebudget many pins along the same path. By default, only selected pins are processed.

-balance true | false

Sets latency targets as if there is zero skew between two blocks, even if the current timing analysis shows otherwise. You should set **-balance** to true whenever some of your clock timing is propagated (not ideal) or estimated, but you have not finished balancing your clocks across the whole chip.

During budgeting, it is possible that the clocks of your designs are not yet balanced. For example, you might have run clock tree synthesis in one of your blocks and not another. Or, you have run clock tree synthesis in each of your blocks, but have not yet balanced the blocks with each other. When you specify **-balance true** to true, the budgeter assumes that clocks are balanced.

If you want some clocks in your design to have a latency that is offset from other clocks, use the **-latency_offset** option of the **set_latency_budget_constraints** command to adjust the interclock balance.

You must specify the **-balance** option together with the **-latency_targets** option.

-ocv_percent *percent*

Adjusts the calculated early and late budget latencies so that they both vary by at least the specified *percent* value from the average.

When you use the **-latency_targets** option, it is possible that on-chip variation (OCV) is not considered. This happens if you are using ideal clocks and you did not supply separate early and late latencies. For example, if both the early and late latency in the design are 10, and you specify **-ocv_percent 5**, the early and late budget latency targets are set to 9.5 and 10.5, respectively.

The **-ocv_percent** option can only be specified together with the **-latency_targets** option.

-ocv_delay *delay*

Adjusts the calculated early and late budget latencies so that they both vary by at least the specified *delay* value from the average.

When you use the **-latency_targets** option, it is possible that OCV is not considered. This happens if you are using ideal clocks

and you did not supply separate early and late latencies. For example, if both the early and late latency in the design are 0, and you specify **-ocv_delay 0.2**, the budget latency targets will be set to -0.2 and 0.2, respectively.

The **-ocv_percent** option can only be specified along with the **-latency_targets** option.

DESCRIPTION

This command automatically assigns new budget constraints for setup delay, boundary parasitics and drive, and/or clock latency. These are controlled by using the **-setup_delay**, **-boundary**, and **-latency_targets** options. If you specify none of these options, setup and boundary will be assumed. See the argument descriptions above for more information.

Note that constraints for hold delay do not depend on whether or not you run the **compute_budget_constraints** command. If you set the **plan.budget.write_hold_budgets** application option to true, hold constraints will automatically be calculated and output by the **write_budgets** command.

The **compute_budget_constraints** can be used to create setup delay constraints on your budget blocks. Running this command is equivalent to calling **set_pin_budget_constraints** directly, except that the constraints are automatically generated. Constraints are generated to distribute slack of top-level paths to the various budget blocks. You can use the **report_budget** command to analyze the results.

Path delays are automatically processed and constraints are set which are the same as those that can be manually set by the following options of the **set_pin_budget_constraints** command:

- -from_percent
- -from_delay
- -to_percent
- -to_delay
- -internal_percent
- -internal_delay

This will affect **set_input_delay** and **set_output_delay** constraints in your final budget.

Block boundaries are automatically processed and constraints are set which are the same as those that can be manually set with the **-early_boundary** and **-late_boundary** options of the **set_pin_budget_constraints** command. This will affect the **set_driving_cell**, **set_load** and **set_max_transition** constraints in your final budget.

By default, **compute_budget_constraints** processes all pins of all budget blocks. Blocks are assigned for budgeting by the **set_budget_options -add_blocks block_name** command. By default, all modes with an active corner are processed, unless you specify **-modes**.

When processing setup delays and boundary constraints, you can limit where **compute_budget_constraints** operates by using one or more of the **-pins**, **-input**, **-output**, **-feedthrough**, or **-slack** options. The **-fanin_cone** and **-fanout_cone** options allow **compute_budget_constraints** to also adjust budgets in the transitive fanin or output of specified pins.

Use the **set_budget_options** command to specify general options to be used when creating timing budgets. To manually set other specific budget parameters, use the **set_pin_budget_constraints**, **set_latency_budget_constraints**, and **set_boundary_budget_constraints** commands.

Use the **write_budgets** command to generate SDC that can be applied to lower level blocks. The **report_budget** command generates useful information about the current budget calculation, allocation, and whether that budget is currently being met. Use the **write_script -include budget** command to save a Tcl script that includes your current settings from this command.

This command returns 1 on success, 0 otherwise.

Multiply Instantiated Blocks

The **compute_budget_constraints** command automatically creates equal budgets for similar pins of multiply instantiated blocks (as directed by the **-same_as_mib** option of the **set_pin_budget_constraints** command). The budget represents the best compromise across the contexts of the multiple instances. However, if it appears that one or more instance pins has a constraint that is impossible to meet, those constraints might be different. You can report the pins with seemingly impossible constraints by using the **report_budget -warning_pins** command.

EXAMPLES

Automatically fill in the budget specification for any pin in any mode where no budget has yet been specified. Be careful not to overwrite previously set boundary budgets.

```
prompt> compute_budget_constraints -unspecified -setup_delay
```

Automatically fill in the budget specification for any pin in any mode where no budget has yet been specified. Also make sure that boundary budgets are automatically created for all pins, regardless of whether or not they have delay budgets.

```
prompt> compute_budget_constraints -unspecified
prompt> compute_budget_constraints -boundary
```

Manually set internal delay budgets for all blocks, then use the **compute_budget_constraints** command to automatically fill in the unspecified feedthrough budgets. Also make sure that boundary budgets are created for all pins.

```
prompt> set_budget_options -add_blocks { core/CPU core/MMU exif }
prompt> set_pin_budget_constraints -all -internal_percent 20
prompt> compute_budget_constraints -unspecified -feedthrough
prompt> compute_budget_constraints -boundary
```

Automatically recalculate all existing budget constraints at the input of core/CPU in the current mode.

```
prompt> compute_budget_constraints -input -pins core/CPU/*
```

Find any pin in "mode2" where the budget is being met by less than 0.05 nanosecond. Assign a new budget to those pins and to pins in the fanin and fanout of those pins.

```
prompt> compute_budget_constraints -modes mode2 -slack 0.05 \
  -fanin_cone -fanout_cone
```

Set up block clock latency targets for early in the flow. Use the current ideal clock specifications, but enforce OCV variability.

```
prompt> compute_budget_constraints -latency_targets actual \
  -balance true -ocv_percent 5
```

Set up block clock latency parameters for early in the flow. Use estimated latency and OCV. Use the **-balance true** option in the assumption that balancing will be done later in the flow.

```
prompt> compute_budget_constraints -latency_targets estimated \
  -balance true
```

Set up block clock latency parameters to exactly match the circuit. The **-balance false** option assumes that the top-level clocks are already balanced.

```
prompt> compute_budget_constraints -latency_targets actual \
  -balance false
```

See the **write_budgets** man page for more examples of the **-latency_targets** option.

SEE ALSO

create_budget_busplan(2)
report_budget(2)
set_boundary_budget_constraints(2)
set_budget_options(2)
set_latency_budget_constraints(2)
set_pin_budget_constraints(2)
set_segment_budget_constraints(2)
synthesize_clock_trunks(2)
write_budgets(2)
write_script(2)
plan.budget.bbt_fixed_delay(3)
plan.budget.estimate_timing_mode(3)
plan.budget.time_with_margins(3)
plan.budget.write_hold_budgets(3)

compute_clock_latency

Updates the network latencies of real and virtual clocks after clock tree synthesis.

SYNTAX

status **compute_clock_latency**
[-verbose]

Data Types

No data types

ARGUMENTS

-verbose

Displays additional columns in the update report to show reference clock, latency calculation method and the number of register clock pins used to calculate the latency.

DESCRIPTION

This command is used primarily to correct your block's timing constraints after clock tree synthesis. Clock tree synthesis inserts clock networks, which changes the latency on your clocks, and the apparent length of the datapaths at your block boundary. The **compute_clock_latency** command can adjust the clock latency on your I/O clocks so that the datapath length is maintained and your boundary constraints are still good. See the example below for more information.

The command calculates and updates the I/O latencies of real and virtual clock objects for all active mode and corner combinations. To calculate and update clock latencies, this command follows the directives set by **set_latency_adjustment_options** command for each mode. If no directives are given, then only the I/O latencies of real clock objects are updated. The latencies are calculated as the median of the clock network latencies at the clock sinks of the clock tree. For ideal clocks, a **set_clock_latency** command is run by the tool on the clock objects. For propagated clocks, I/O path latency is set internally on the clock objects. The I/O path latency is used in timing the input/output ports. The command updates latencies for all the active mode and corner combinations.

Note that **compute_clock_latency** is run automatically by the tool as part of integrated CTS and optimization flows. So it is important to use the **set_latency_adjustment_options** command, even if you don't intend to call **compute_clock_latency** explicitly.

Multicorner-Multimode Support

This command works on all active scenarios.

EXAMPLES

This example adjusts the latency on the specified I/O clocks. First, you must associate your I/O clocks with your internal clocks by using the **set_latency_adjustment_options** command. Then, when you run **compute_clock_latency**, the latency of the I/O clock will be set to match your internal clock. Notice that the **write_io_constraints** command can set latency adjustment options with **set_latency_adjustment_options** in the budget script.

```
prompt> set_latency_adjustment_options \
  -reference_clock my_internal_clock \
  -clocks_to_update my_io_clock
prompt> synthesize_clock_trees
prompt> compute_clock_latency

Mode:mode2          Latency(early)   Latency(late)
Clock      Updated  rise   fall   rise   fall   Corner
-------------------------------------------------------------
clock      Yes    0.9336    --  0.9336    --    c2
clock_dup  No       --     --     --     --    --
clock_dup_2 Yes    1.6785    --  1.7258    --    c2
vclk1      Yes    0.9572    --  0.9572    --    c2
vclk2      Yes    0.9336    --  0.9336    --    c2

Mode:mode1          Latency(early)   Latency(late)
Clock      Updated  rise   fall   rise   fall   Corner
-------------------------------------------------------------
clock      Yes    0.9336    --  0.9336    --    c1
clock_dup  No       --     --     --     --    --
clock_dup_2 Yes    1.6785    --  1.7258    --    c1
vclk1      Yes    0.9572    --  0.9572    --    c1
vclk2      Yes    0.9336    --  0.9336    --    c1
```

The following example generates a report with additional information by using the **-verbose** option.

```
prompt> compute_clock_latency -verbose

Mode:default        Latency(early)   Latency(late)
Clock      Updated  rise   fall   rise   fall   Corner  Exclude
  Ref Clock   Computed using #regs
-------------------------------------------------------------------------
clock      Yes    0.9336    --  0.9336    --    default  No
  -self-     io pair regs     4
clock_dup  Yes    0.9336    --  0.9336    --    default  No
  -self-     all boundary regs 8
vclk1      No       --     --     --     --    --     No
  --        --             0
vclk2      No       --     --     --     --    --     No
  --        --             0
```

SEE ALSO

set_latency_adjustment_options(2)

write_io_constraints(2)

compute_dff_connections

Traces connections in the current netlist. The results are used by the Data Flow Flylines tool in the GUI.

SYNTAX

status **compute_dff_connections**
[-max_reg *register_count*]
[-max_gate *gate_count*]
[-max_fanout *fanout_count*]
[-include_blocks *list_of_designs*]
[-retrace_blocks *list_of_designs*]
[-host_options *host_option_name*]
[-work_dir *working_directory*]
[-dont_include_abstracts]

Data Types

register_count int
gate_count int
fanout_count int
list_of_designs collection
host_option_name string
working_directory string

ARGUMENTS

-max_reg *register_count*

Specifies the maximum number of registers a path can contain and still be stored. Paths containing more than this number of registers are not stored for display. By default, this value is 3.

-max_gate *gate_count*

Specifies the maximum number of total gates a path can contain and still be stored. Gates are considered combinational cells that are not buffers or inverters. Paths containing more than this number of gates are not stored for display. By default, this value is 20.

-max_fanout *fanout_count*

Specifies the maximum fanout permitted for any net found while tracing the path. If a net with fanout greater than this limit is encountered, the path is not stored for display. By default, this value is 64.

-include_blocks *list_of_designs*

Specifies the abstract views for which to load previously saved trace data. The abstract view must be currently loaded or available in the library, and the **-max_reg**, **-max_gate**, and **-max_fanout** option values must be consistent with the previously saved data. If no compatible trace data file exists for the abstract view, you must also specify the *-retrace_blocks* option. By default, the commands loads all design views and abstract views for blocks returned by the *get_designs* command.

-retrace_blocks *list_of_designs*

Specifies a collection of abstract designs and physical blocks for which to retrace the netlist. *list_of_designs* is a set or subset of designs returned from the *get_designs* command. The specified designs are opened in design view and traced using the current tracer settings. The trace data is saved into a tracer data file that can be loaded by the Data Flow Flylines component in the GUI. Any design specified for this option must also be specified for the *-include_blocks* option. Note: after tracing any designs in this manner, the main design is reloaded and relinked, and all existing object collections become invalid and must be re-created.

-host_options *host_option_name*

Specifies the name of the host option to use for distributed processing. By default, this is empty and distributed processing is not used. See also *set_host_options*.

-work_dir *working_directory*

Specifies the directory in which to write the tracer files. By default, the command writes to a directory named "work_dir".

-dont_include_abstracts

Ignores abstracts and traces only in design views.

DESCRIPTION

This command invokes the Data Flow Flylines tracer. The tracer traces through the current netlist and stores paths that meet the specified constraints. These paths can then be displayed using the Data Flow Flylines GUI component. Several options are available to restrict the types of paths stored and limit the runtime required by the tracer (which might take several minutes for a larger design).

This command uses the settings specified by the *create_dff_trace_filters* command to stop tracing when any of the specified filter pins or filter nets are encountered during tracing.

If the current design contains abstract blocks, they can still be included in the tracer results. By specifying the *-include_blocks* and *-retrace_blocks* (if required) options, the specified abstract view designs can be opened in design view, either in the current session or a distributed session, depending on the current distributed processing settings. For more information, see the man pages for **set_host_options** and **run_block_script**.

After saving tracer data files for the abstract designs, these sessions are closed and the saved tracer data files can be merged into the main tracer results. This only applies to abstract designs, and designs in design view are always automatically included in the main tracer results.

EXAMPLES

The following example traces the current design using these constraints: paths with a register depth of 2 or less, a maximum gate count of 25 or less, and a maximum fanout of 64, which is default. The Data Flow Flylines tool can be used to view the results in the GUI.

```
prompt> compute_dff_connections -max_reg 2 -max_gate 25
```

The following example traces the current design, but also forces a retrace of the ABS1 abstract block and includes the ABS1 and ABS2 abstract blocks in the main trace results.

```
prompt> set a1 [get_designs {ABS1 ABS2}]
{ABS1 ABS2}
prompt> set a2 [get_designs {ABS1}]
{ABS1}
prompt> compute_dff_connections -include_blocks $a1 -retrace_blocks $a2 -host_options opt1
found instance TOP/I_ABS1 for design ABS1
Block run directory is /.../work_dir/compute_dff_connections
Submitting job for block ABS1 ...
All tasks created.
current lib is /.../curlib
current block is MAIN.design
save all libs ...
Saving all libraries...
1
close all libs.
Closing all libraries...
1
Running distributed block jobs ...
(all scripts and log files are under /.../work_dir/compute_dfa_connections)
Worker_msg: block ABS1 starts : Wed Nov 12 18:40:09 2014
Worker_msg: block ABS1 log :
  /.../work_dir/compute_dfa_connections/ABS1/ABS1.22766.log :
  Wed Nov 12 18:40:09 2014
Block Ref ABS1
  Wall clock time spent pending : 00:00:06.08
  Wall clock time spent running : 00:00:05.06
Distributed run summary
  Wall clock time (total)           : 00:00:41.19
  Longest running block ref         : ABS1
  Wall clock time lost due to pending : 00:00:06.08
Distributed run end.
open lib /.../curlib
Information: Loading library file '/.../curlib' (FILE-007)
Information: Loading library file '/.../MAIN.nlib' (FILE-007)
{curlib}
open block MAIN
Opening block 'curlib:MAIN.design'
{curlib:MAIN.design{
Using libraries: curlib MAIN
Visiting block curlib:MAIN.design
Visiting block curlib:ABS2.abstract
Visiting block curlib:ABS1.abstract
Expanding block curlib:MAIN.design
cell TOP/I_ABS2 is referencing to ABSTRACT view.
cell TOP/I_ABS2 is referencing to ABSTRACT view.
Design 'MAIN' was successfully linked.
1
current block is MAIN.design
Loading abstract design : ABS2
Loading abstract design : ABS1
1
```

SEE ALSO

create_abstract(2)
create_dff_trace_filters(2)
merge_abstract(2)
remove_dff_trace_filters(2)
report_host_options(2)
run_block_script(2)
set_host_options(2)
write_dff_trace_filters(2)

compute_infeasible_path_overrides

checks infeasible path margins.

SYNTAX

status **compute_infeasible_path_overrides**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-group *group_list*]
[-apply_override_constraints]
[-output *file_name*]
[-nworst *paths_per_endpoint_per_path_group*]
[-max_paths *max_path_count_per_path_group*]
[-verbose]

Data Types

mode_list	list
corner_list	list
scenario_list	list
group_list	list
file_name	string

ARGUMENTS

-modes *mode_list*

Specifies the scenarios to check infeasibility. If this option is given, all scenarios of the given modes are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-corners *corner_list*

Specifies the scenarios to check infeasibility. If this option is given, all scenarios of the given corners and the current mode is used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-scenarios *scenario_list*

Specifies the scenarios to check infeasibility. The **-modes** or **-corners** option must not be given with this option.

-group *group_list*

Specify the list of path group names for checking infeasibility. Default is all path groups.

-apply_override_constraints

This option generates and applies path margins. Added path margins are all commented with SYNOPSYS_INFEASIBLE_PATH_OVERRIDE.

-output *file_name*

Specifies the name of the generated file containing timing exceptions to resolve infeasibility.

-nworst *paths_per_endpoint*

Specifies the number of paths to search per endpoint. Value is applied to each path group. The default value is 1000.

-max_paths *max_path_count*

Specifies the number of paths to report. The default value is 10000. The max_paths value is applied to each path group.

-verbose

This option generate arrival time at startpoint and required time at endpoint along with infeasibility (arrival time at startpoint - required time at endpoint)

DESCRIPTION

This command checks timing paths infeasibility.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

this example checks infeasibility in scneario s1

prompt> **compute_infeasible_path_overrides -scenario s1**

This example checks infeasibility in current scenario and applies set_path_margin exceptions to resolve infeasibility.

prompt> **compute_infeasible_path_overrides -apply_override_constraints**

SEE ALSO

remove_path_margin(2)

compute_polygons

Performs a boolean geometric operation and returns the result as a geo_mask.

SYNTAX

collection **compute_polygons**
-operation AND | OR | XOR | NOT
-objects1 *object_list*
-objects2 *object_list*

Data Types

object_list collection

ARGUMENTS

-operation AND | OR | XOR | NOT

Specifies the Boolean operation to perform:

AND Get the region covered by both *objects1* and *objects2*
OR Get the region covered by either *objects1* or *objects2*
XOR Get the region covered by *objects1* or by *objects2* but not both
NOT Get the region covered by *objects1* but not by *objects2*
TOUCHING Get the region covered by the poly_rects in *objects1* that abuts or overlaps with one or more poly_rects in *objects2*
INTERSECT Get the region covered by the poly_rects in *objects1* that overlaps with one or more poly_rects in *objects2*

This is a required option.

-objects1 *object_list*

Specifies the objects to be used to define the geometric region of the first argument of the operation. Objects can be a heterogeneous collection of poly_rects, geo_masks, shapes, layers, and other physical objects.

In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

This is a required option.

-objects2 *object_list*

Specifies the objects to be used to define the geometric region of the second argument of the operation, and has the same form and functionality as the *-objects1* argument.

This is a required option.

DESCRIPTION

This command performs the specified Boolean operation on *objects1* and *objects2*, and returns a collection that contains the result as a geo_mask object.

EXAMPLES

The following example returns a geo_mask with a region that is the union of two shapes.

```
prompt> compute_polygons -objects1 [get_shapes RECT_0] -operation AND \
    -objects2 [get_shapes RECT_1]
```

The following example returns a geo_mask with the region composed of the shapes on a layer, except for the areas of shapes that lie within a rectangle.

```
prompt> compute_polygons -objects1 [get_layers M1] -operation NOT \
    -objects2 [create_poly_rect -boundary {{0 0} {200 100}}]
```

SEE ALSO

compute_area(2)
copy_to_layer(2)
create_geo_mask(2)
create_poly_rect(2)
split_polygons(2)
resize_polygons(2)
transform_polygons(2)

connect_freeze_silicon_tie_cells

Connect target pins to tie cells in freeze silicon mode.

SYNTAX

status **connect_freeze_silicon_tie_cells**
[-cells *cells*]
[-max_fanout *number*]
[-max_wire_length *distance*]
[-tie_high_lib_cell *lib_cell*]
[-tie_low_lib_cell *lib_cell*]

Data Types

cells collection
lib_cell collection
number integer
distance distance

ARGUMENTS

-cells *cells*

Specifies the target cells to be connected to tie cells. These cells must have pins already connected to tie-low nets or tie-high nets. This option is optional, and if this option is not specified, target cells include all cells which have pins already connected to tie nets in the design.

-max_fanout *number*

Specifies the maximum number of pins that can be driven by a single tie-high or tie-low cell. If you do not specify this option, the maximum fanout is 8.

-max_wire_length *distance*

Specifies the maximum Manhattan wire length in microns from a tie-high or tie-low cell to each driven cell pin. If you do not specify this option, the maximum wire length is 100.

-tie_high_lib_cell *lib_cell*

Specifies the reference cell type of tie cells to be used for tie high connections. If not specified, all types of tie high cell can be used for tie high connections If you use this option, you must also use the **-tie_low_lib_cell** option.

-tie_low_lib_cell *lib_cell*

Specifies the reference cell type of tie cells to be used for tie low connections. If not specified, all types of tie low cell can be used for tie low connections If you use this option, you must also use the **-tie_high_lib_cell** option.

DESCRIPTION

This command connect pins that are already connected to tie nets to tie cells in freeze silicon mode. If users specify the option **-cells** , only pins of these cells that are connected to tie nets will be connected to tie cells. And these pins are connected to surrounding tie cells as close as possible.

This command can work in freeze silicon mode, and spare tie cells can be used to create new tie cells in freeze silicon mode if needed.

The pins that already connected to tie nets are only connected to tie cells in the same voltage area of these pins by this command.

This command support spare tie cells and non-spare tie cells. Non-spare tie cells can only be connected to pins in the same hierarchy. But for a spare tie cell, it could be moved to another hierarchy (swap out and a new tie cell is created) and connected to pins in that hierarchy.

Cells must be placed before the command, include tie cells and cells that need to be connected to tie cells. And cells inside sub block are also not supported.

Users can specify maximum fanout (the maximum number of pins that can be driven by a single tie cell) and maximum wire length (the maximum Manhattan wire length between a pin and its tie cell driver) constraints.

Users can specify tie high and tie low lib cells, and then only tie cells of the specified lib cells can be used for tie connections. If not specified, all tie-high and tie-low cells can be used for tie connections.

When this command succeeds, the toll will report a summary like this:

```
****************************************
Report : ECO connect_freeze_silicon_tie_cells Summary
 ...

****************************************
Total 168 pins are connected to tie cells.

Voltage Area DEFAULT_VA:
   Connect    6 pins to tie-low cells.
   Connect    4 pins to tie-high cells.

Voltage Area pd2:
   Connect  102 pins to tie-low cells.
   Connect   34 pins to tie-high cells.

Voltage Area pd3:
   Connect    6 pins to tie-low cells.
   Connect    2 pins to tie-high cells.

Voltage Area pd4:
   Connect    8 pins to tie-low cells.
   Connect    6 pins to tie-high cells.
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

In the following example, there is no option specified, and all pins that already connected to tie nets will be connected to tie cells.

```
prompt> connect_freeze_silicon_tie_cells

****************************************
Report : ECO connect_freeze_silicon_tie_cells Summary
 ...

****************************************
Total 22 pins are connected to tie cells.

Voltage Area pd3:
   Connect    6 pins to tie-low cells.
   Connect    2 pins to tie-high cells.

Voltage Area pd4:
   Connect    8 pins to tie-low cells.
   Connect    6 pins to tie-high cells.
1
```

In the following example, only pins of the specified cells are connected to tie cells if they are already connected to tie nets.

```
prompt> connect_freeze_silicon_tie_cells -cells {alu/EcoCell*}
```

In the following example, the options *-max_fanout* and *-max_wire_length* are specified, and the command will follow these constrains when connect pins to tie cells.

```
prompt> connect_freeze_silicon_tie_cells -max_fanout 5 -max_wire_length 20
```

In the following example, the options *-tie_high_lib_cell* and *-tie_low_lib_cell* are specified, and the command will only use tie cells of the specified lib cells for tie connections.

```
prompt> connect_freeze_silicon_tie_cells -tie_high_lib_cell tcbn90ghvt/TIEHHVT \
   -tie_low_lib_cell tcbn90ghvt/TIELHVT
```

SEE ALSO

add_spare_cells(2)
place_eco_cells(2)

connect_logic_net

Connects a logic net to logic ports.

SYNTAX

status **connect_logic_net**
net_name
[-ports *port_list*]
[-reconnect]

Data Types

net_name string
port_list list

ARGUMENTS

net_name

Specifies the net name.

-ports *port_list*

Specifies the logic ports connected to the net. The ports must be on the interface of the active scope or on the design elements that are located in the active scope and its descendants.

-reconnect

Allows a port that is already connected to a net to be disconnected from the existing net and connected to the newly specified net.

DESCRIPTION

The connect_logic_net command connects a logic (non-PG) net to the specified logic ports. The net is propagated through implicitly created ports and nets throughout the logic hierarchy in the descendant subtree of the active UPF scope as required to support connections created by connect_logic_net. A unique global driver needs to be present among the logic net and logic ports if there exists a logic port not in the same hierarchy as any segment of the given net in the active UPF scope. This command will not support multiple driver nets. That is, it will not implement connection if there are multiple drivers specified, or the net has multiple drivers.

On the path from a specified logic port to the logic net:

-

no internal port can be created on a power domain boundary when establishing a new connection;

-

there cannot be a different net already connected to the logic port.

The **-reconnect** option disconnects any existing net from the specified port and connects the newly specified net to all the ports present in the port_list.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

connect_net(2)
create_logic_port(2)
create_logic_net(2)

connect_net

Connects a net or scalar nets of net_bus to port, pins and scalar ports of port_bus.

SYNTAX

int **connect_net**
[-design *design*]
-net *net*
-port_name *port_name*
net
terms

Data Types

design string
net list
port_name string
terms list

ARGUMENTS

-design *design*

Specifies the design in which to connect the nets. If no design is specified, the nets are connected in the current design.

-net *net*

Specifies the net or net_bus to connect. It can be a name or collection. The net can be a scalar (single-bit) net or net_bus and must exist in the current design. If net_bus is specified, then terms must contain the same width port_bus or length of the scalar port array list should be same as bus width of the net_bus.

net

Specifies the net or net_bus to connect. It can be a name or collection. The net can be a scalar (single-bit) net or net_bus and must exist in the current design. If net_bus is specified, then terms must contain the same width port_bus or length of the scalar port array list should be same as bus width of the net_bus. This positional argument is provided for compatibility with other tools.

-port_name *port_name*

Specifies the base name to use as the name of new ports that are created on subdesigns to make the connection.

terms

Specifies a list of ports, port_bus and pins to which the net is to be connected. The ports and pins must exist in the current design. If a specified port, scalar port of port_bus or pin is already connected, the command issues an error message.

DESCRIPTION

This command connects a net or scalar net of the net_bus to the specified ports, port_bus and pins, possibly across several levels of hierarchy. port_bus connections across hierarchies is not supported. A net can be connected to many ports and pins, however you cannot connect a port or pin to more than one net. Ports and nets are created as needed to cross hierarchy.

If a port or pin cannot be connected, the port or pin is skipped. The remaining ports and pins in the *terms* are connected.

The number of connected ports and pins are returned.

To disconnect objects on a net, use the **disconnect_net** command. To display ports and pins on a net, use the **all_connected** command.

EXAMPLES

The following example connects a net named *NET0*. The **all_connected** command returns the objects connected to *NET0*.

```
prompt> connect_net -net NET0 [get_ports { A1 A2 } ]


prompt> connect_net -net NET0 [get_pins U1/A]


prompt> all_connected NET0


{"A1", "A2", "U1/A"}
```

The following example connects a net_bus named *N[7:0]* to port_bus named *P[7:0]* so that N[7] is connected to P[7], N[6] is connected to P[6] ..,N[0] to P[0].

```
prompt> connect_net -net N P
8


# The nets can be explicitly mentioned as follows
prompt> connect_net -net [get_net_buses N] [get_port_buses P]
8


prompt> all_connected N[1]
{"P[1]"}
```

SEE ALSO

all_connected(2)
disconnect_net(2)

connect_pg_net

Connects power, ground, and tie-off pins to the specified power and ground nets. The command supports both an automatic mode and a manual mode. The default is automatic mode. In automatic mode, the tool derives the power and ground nets from the power domain connections. Automatic mode requires committed UPF descriptions and logic libraries with power and ground pins. In manual mode, you specify the power and ground nets. In automatic mode, the tool reports PG net and PG pin connections status at the end.

SYNTAX

status **connect_pg_net**
[-automatic]
[-all_blocks]
[-block *block_list*]
[-design *design*]
[-net *net*]
[*port_pin_list*]
[-create_nets_only]
[-verbose]

Data Types

block_list list
design collection
net list
port_pin_list list

ARGUMENTS

-automatic

Specifies automatic mode. This option cannot be used with **-net** or port_pin_list options.

-all_blocks

Traverses all physical blocks during auto PG connection. This option cannot be used with **-block**, **-net** or port_pin_list options.

-block *block_list*

Traverses the current design, as well as the physical blocks specified in *block_list* during auto PG connection. Blocks in *block_list* that are not in current design are ignored. This option cannot be used with **-all_blocks**, **-net** or port_pin_list options.

-design *design*

Specifies the design in which to connect power and ground nets. If no design is specified, the power and ground nets are connected in the current design.

-net *net*

Specifies the PG net to connect. The *net* argument can be a name or collection. The net must be a scalar (single-bit) PG net and must exist in the current design.

port_pin_list

Specifies a list of ports and pins to which the net is to be connected. The ports and pins must exist in the current design. If a specified port or pin is already connected, the command disconnects the existing connection and makes a new connection.

-create_nets_only

Will only automatically create topmost PG nets based on power constraints, even the design contains unmapped cells.

-verbose

Prints the details of the changes to the power, ground, and tie-off connections.

DESCRIPTION

This command creates the power and ground network for your design by connecting the power, ground, and tie-off pins to the power and ground nets in your design.

The command supports two modes: automatic mode and manual mode.

In automatic mode, the tool derives the power and ground nets from the power domain connections. UPF must be committed before running **connect_pg_net** in automatic mode. In addition, automatic mode requires logic libraries with power and ground pins. In automatic mode, the tool creates the power and ground nets if they do not already exist.

In manual mode, use **-net** option to specify the power and ground nets. Using **-net** option invokes the tool in manual mode. You can use manual mode for single-voltage or multivoltage designs.

EXAMPLES

The following example uses the **connect_pg_net** command in automatic mode to connect all unconnected power and ground pins based on the power domain connections.

```
prompt> connect_pg_net -automatic

****************************************
Report : Power/Ground Connection Summary
Design : TOP
Version: N-2017.09-SP6-VAL
Date   : Tue Jun 19 02:45:44 2018
****************************************
P/G net name                          P/G pin count(previous/current)
--------------------------------------------------------------------------------
Power net VDD1SD                          0/8
Power net VDD3                           0/6
Power net VDD2                           0/6
Power net VDD1                           0/0
Unconnected nwell pins                    20
```

```
Ground net VSS                          0/20
Unconnected pwell pins                    20
--------------------------------------------------------------------------------
Information: connections of 40 power/ground pin(s) are created or changed.
```

The following example uses the **connect_pg_net** command in manual mode to connect power pin vdd_io on cell I_PAD1 to power net VDD_IO. The command also connects power pin vdd_core on the same cell to power net VDD_CORE.

```
prompt> connect_pg_net -net VDD_IO   [get_pins I_PAD1/vdd_io]
prompt> connect_pg_net -net VDD_CORE [get_pins I_PAD1/vdd_core]
```

SEE ALSO

resolve_pg_nets(2)
commit_upf(2)

connect_pg_via_ladders

Connect via ladders to closest PG straps.

SYNTAX

status **connect_pg_via_ladders**
[-nets *pg_nets*]
[-cells *instances*]
[-pins *pins*]
[-target_layer string]
[-target_shapes *shapes*]
[-ignore_drc]
[-tag string]
[-via_masters *via_masters*]
[-shape_use *shape_use*]
[-number_of_targets integer]
[-max_extension *distance*]

Data Types

pg_nets string or list of string
instances collection
shape_use string
pins collection
shapes collection
via_masters string or list of string
distance float

ARGUMENTS

-nets *pg_nets*

Specifies the power ground nets to be connected. -nets must be used together with -cells. -nets is exclusive with -pins.

-cells *instances*

Specifies the instances to be connected. Tool only makes connection of instances having via ladders created for corresponding pins. -cells must be used together with -nets. -cells is exclusive with -pins.

-pins *pins*

Specified the pins to be connected. Tool only makes connection of pins with which via ladders associated. -pins is exclusive with -cells and -pins.

-target_layer string

Specifies the target layer on which power straps should be connected.

-target_shapes *shapes*

Specifies a collection of shapes which the via ladders should be connected to.

-ignore_drc

Specifies if drc should be checked for the connection.

-tag string

Specifies the tag info for created PG shapes/vias.

-shape_use *shape_use*

Specified the shape use of created PG shapes/vias. By default it is stripe

-via_masters *via_masters*

Specifies the via masters to be used for via ladder connection. If not specified, default contact code will be used.

-number_of_targets integer

Specifies the number of straps to which the via ladder should connect. If not specified, only one connection will be made for each via ladder.

-max_extension distance

Specifies the maximum extension of the via ladder to make connection. If not specified, the range is not limited. If max extension is specified, the command will try all the targets within the range, in the order of distance, until the number of connections specified by -number_of_targets is met. If max extension is not specified, the command only tries the number of targets specified by -number_of_targets, no matter if a connection can be made successfully or not.

DESCRIPTION

This command tries to connect top level shape of via ladders to existing PG mesh. It will only use straight connections. As a result, there might be via ladders left open.

EXAMPLES

The following example make via ladders of secondary pg pins vdd_ron to PG straps on M5.

prompt> **connect_pg_via_ladders -nets vdd_ron -cells $cell_collection -target_layer M5**

SEE ALSO

compile_pg(2)
route_group(2)

connect_pins

Connects a set of pins together across hierarchy.

SYNTAX

int **connect_pins**
-incremental
-non_incremental
-driver *pin*
-port_name *port_name*
loads

Data Types

pin collection
port_name string
loads list

ARGUMENTS

-incremental

Make the operation incremental. The driver pin must be connected to a net otherwise the option does not have any effect. Any existing loads on the net connected to the driver pin are preserved. Please note that connect_pins behavior defaults to -incremental usage, therefore explicit use of -incremental is not required.

-driver *pin*

Specifies the driving pin of the network. It can be a name or collection. The pin must exist in the current design.

-port_name *port_name*

Specifies the base name to use as the name of new ports that are created on subdesigns to make the connection.

loads

Specifies a list of ports and pins to which the driving pin is to be connected. The ports and pins must exist in the current design.

-non_incremental

Make the operation non-incremental. If a specified load port or pin is already connected to something else, it will be disconnected and then connected to the driving pin.

DESCRIPTION

This command connects a set of pins together, possibly across several levels of hierarchy. Ports and nets are created as needed to cross hierarchy. Existing nets and ports may be reused to complete a connection.

The number of connected load ports and pins are returned.

To disconnect objects on a net, use the **disconnect_net** command. To display ports and pins on a net, use the **all_connected** command.

EXAMPLES

The following example connects a the driving pin u20/ZN to a number of input pins across several levels of hierarchy. The **getPins** command then returns the list of pins connected across the hierarchy.

```
prompt> connect_pins -driver u20/ZN {u21/I u1/u22/I u2/u23/I u4/u2/u24/I u2/u2/u2/u25/I}}

5

prompt> get_pins -leaf -of_objects [get_nets -of [get_pins u20/ZN]]

u1/u22/I u2/u2/u2/u25/I u2/u23/I u20/ZN u21/I u4/u2/u24/I
```

SEE ALSO

all_connected(2)
connect_net(2)
disconnect_net(2)
get_nets(2)
get_pins(2)

connect_power_switch

Connects the pins of the power switch cells in the design.

SYNTAX

int **connect_power_switch**
-source *object*
-port_name *port_name*
-mode hfn | daisy | fishbone | clustering | shortest
[-ack_out *object*]
[-ack_port_name *port_name*]
[-direction <horizontal | vertical>*]
[-start_point <lower_left | upper_left | lower_right | upper_right>*]
[-ring_direction clockwise | counter_clockwise]
[-lib_pin *library_pin_names*]
[-voltage_area *voltage_areas*]
[-object_list *objects*]
[-keep_order]
[-fine_grained_lib_pins *library_pin_pairs*]
[-connect_sleep_and_ack_in_shared_domains]

Data Types

object string
port_name string
library_pin_names list
voltage_areas list
objects collection
library_pin_pairs list

ARGUMENTS

-source *object*

Specifies the pin or port from which to connect. The direction of the pin or port cannot be inout. In clustering mode, this option supports a collection of pins and ports.

-port_name *port_name*

Specifies the base name to use as names of switch ports when a new port is created on the subdesigns to make the connection. It is a required option.

-mode hfn | **daisy** | **fishbone** | **clustering** | **shortest**

Specifies whether the switch connections are to be made in high fanout, daisy , fishbone, clustering, or shortest style. It is a

required option.

-ack_out *object*

Specifies the pin or port that is to be used as acknowledge-out connection. It is optional and is only allowed with **-mode daisy** or **-mode fishbone**.

-ack_port_name *port_name*

Specifies the base name to use as names of acknowledge ports when a new port is created. This is required with the **-ack_out** option in **-mode daisy** or **-mode fishbone**.

-direction <horizontal | vertical>*

Specifies the direction for the daisy and fishbone connections. Requires **-mode daisy** or **-mode fishbone**. It is used only when the tool tries to infer the daisy chain direction based on the instantiated switch cell locations. This option can also be used to specify the direction of connections for branches connected to the main trunk in **-mode fishbone**.

If **-mode daisy** and multiple **-object_list** are specified, **-direction** can take a list of values, one for each object list, to specify the connection direction for power switch cells in each object list. If there are more **-object_list** than the number of values listed in **-direction**, the last value in **-direction** will be used as the connection direction for the rest of the object lists.

-start_point lower_left | upper_left | lower_right | upper_right

Specifies the starting point for the daisy mode connections. Requires **-mode daisy** or **-mode fishbone**. It is used to infer the startpoint for daisy or fishbone connections. This is helpful in connecting the source pin to the nearest starting switch cell. For fishbone connections, this option determines the starting point for the trunk of fishbone connections. This option can only be used with **-direction**. This is not a required option.

If **-mode daisy** and multiple **-object_list** are specified, **-start_point** can take a list of values, one for each object list, to specify the starting position for power switch cells in each object list. If there are more **-object_list** than the number of values listed in **-start_point**, the last value in **-start_point** will be used as the startng point for the rest of the object lists.

-ring_direction clockwise | counter_clockwise

Specifies the direction for Ring style switch placement. Now this option can support switches inserted using **create_power_switch_ring** or by other non-SNPS tool. When this option is specified with **-direction**, the **-direction** will be ignored. And this option supports **daisy** mode only.

-lib_pin *library_pin_names*

Specifies the list of library input and output pins that should be used in all the connections modes. This is a required option for switch cells that have multiple switch and acknowledge pins. The command issues an error message if a switch cell with multiple input or output is used but does not have its library pins specified in this option.

-voltage_area *voltage_areas*

Specifies the voltage areas. The command tries to collect the switches in the specified voltage area and connect them. This command supports both single shape and disjoint voltage areas.

-object_list *objects*

Specifies the list of cells on which to perform the connections. If **-mode daisy** or **-mode fishbone** is specified, the tool will automatically sort the object list to do the connections.

If **-mode daisy** is specified, multiple **-object_list** are allowed. In this setting, the relative order of power switch cells in different **-object_list** will be maintained while power switch cells in the same **-object_list** will be sorted. For example, if sw1 is in the first **-object_list** and sw2 is in the second **-object_list**, sw1 will always appear before sw2 in the final power switch chain.

-keep_order

Specifies whether to keep the order of power switch cells you specify with -object_list option. It only works for **-mode daisy**. You cannot specify -keep_order for **-mode hfn** or **-mode fishbone**. Without -keep_order option, tool will sort power switch cells automatically.

-fine_grained_lib_pins ***library_pin_pairs***

Specifies the list of sleep and ack library pin pairs of fine-grained macro cells. This option can only be used with **-connect_sleep_and_ack_in_shared_domains**.

-connect_sleep_and_ack_in_shared_domains

When this command option is specified, the tool will connect the sleep input and ack output of other shared power domains if they are unconnected. The ack output specified in the '-ack_port' command option will be connected to and drive the control inputs and ack outputs of other shared power domains in daisy-chain fashion.

DESCRIPTION

The **connect_power_switch** command connects control and ack_out pins of power switch cells that are instantiated and placed in the design. Connection modes and ordering for the switches are specified by options. The command can identify the switch cells in the design based on Liberty syntax and can connect them in either high-fanout, daisy-chain or fishbone style.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples showcase the command usage:

```
prompt> connect_power_switch \
   -source enable \
   -port_name enable_va_0 \
   -mode daisy \
   -direction horizontal \
   -voltage_area voltage_area_2
1
prompt> connect_power_switch \
   -source enable \
   -port_name enable_va_0 \
   -mode daisy \
   -direction horizontal \
   -object_list $switch_list1 \
   -start_point lower_right
1
prompt> connect_power_switch \
   -source enable \
   -port_name enable_va_0 \
   -mode fishbone \
   -direction horizontal \
   -object_list $switch_list1
```

```
1
prompt> connect_power_switch \
   -source enable \
   -port_name enable_va_0 \
   -mode daisy \
   -direction horizontal \
   -lib_pin {lib1/header32/SLEEP1 lib1/header32/SLEEPOUT1}
   -object_list {Header_switch1 Header_switch100}
1
prompt> connect_power_switch \
   -source [get_pins {BUF1/Y BUF2/Y}] \
   -port_name enable_va_0 \
   -mode clustering \
   -object_list [get_cells {Header_switch*}]
1
prompt> connect_power_switch \
   -source mult_on \
   -port_name mult_on_test \
   -mode daisy \
   -ring_direction clockwise \
   -voltage_area [get_voltage_areas InstDecode/PD]
1
prompt> connect_power_switch \
   -source enable \
   -port_name enable_va_0 \
   -ack_out enable_out \
   -ack_port_name enable_out_va_0 \
   -mode daisy \
   -direction {horizontal vertical horizontal} \
   -start_point {lower_right lower_left lower_right} \
   -object_list $switch_list1 \
   -object_list $switch_list2 \
   -object_list $switch_list3
1
```

SEE ALSO

create_power_switch(2)
create_voltage_area(2)
create_power_switch_array(2)
create_power_switch_ring(2)

connect_supply_net

Connects the supply net to the specified supply ports and pins.

SYNTAX

status **connect_supply_net**
supply_net_name
-ports *list*
-vct *vct_name*

Data Types

supply_net_name string
list list
vct_name string

ARGUMENTS

supply_net_name

Specifies the name of the supply net to connect. If a supply net with the specified name does not exist in the current scope, the supply net cannot be connected. This is a required option and must be specified.

-ports *list*

Specifies which supply ports and pins to connect to the supply net. The name is a hierarchical name.

-vct *vct_name*

Specifies the value conversion table (VCT) to be used in this connection. A value conversion table (VCT) defining how values are mapped from UPF to an HDL model or from the HDL model to UPF.

The tool supports the following 14 predefined VCT names: UPF2VHDL_SL, UPF_GNDZERO2VHDL_SL, UPF2SV_LOGIC, UPF_GNDZERO2SV_LOGIC, VHDL_TIED_HI, SV_TIED_HI, VHDL_TIED_LO, SV_TIED_LO, VHDL_SL2UPF, VHDL_SL2UPF_GNDZERO, SV_LOGIC2UPF, SV_LOGIC2UPF_GNDZERO, SV_LOGIC2UPF_MD, and SV_LOGIC2UPF_GNDZERO_MD.

The tool also support user defined vcts created by **create_hdl2upf_vct** and *create_upf2hdl_vct*.

DESCRIPTION

The **connect_supply_net** command connects a supply net to any supply port or pin. The command overrides (has higher

precedence than) any auto-connection semantics that might otherwise apply. If a design element is not connected explicitly to any supply net using the **connect_supply_net** command, it will share the primary power or ground supply net with the power domain that it belongs to.

The supply net must be created in the same power domain as the supply port, before they can be connected. The instance that contains the pin must also be in the extent of the same power_domain. If the specified supply net or port or pin is not in the extent of same power domain, this command fails.

This command returns 1 on success, returns 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example connects a supply net VSS with supply port VSS:

```
prompt> create_power_domain PD1 -elements INST_1
PD1
prompt> create_supply_net VSS -domain PD1
VSS
prompt> create_supply_port VSS -domain PD1
VSS
prompt> connect_supply_net VSS -ports VSS
1
```

SEE ALSO

create_supply_net(2)
get_related_supply_nets(2)
get_supply_nets(2)
report_supply_nets(2)
create_upf2hdl_vct(2)
create_hdl2upf_vct(2)

convert_aocv_pocv

Converts AOCV derating factor tables to POCV coefficient tables.

SYNTAX

status **convert_aocv_pocv**
[-depth *depth_row*]
[-corners *corner_list*]

Data Types

depth_row integer
corner_list list

ARGUMENTS

-depth *depth_row*

Specifies to use the *depth_row* entry in the AOCV table to do the conversion. By default, the first entry of the depth row in the AOCV table is used.

-corners *corner_list*

Indicates the list of corners for which the AOCV derate factors are to be converted and the converted POCV coefficients are to be applied. If this option is not provided, the corner associated with the current scenario is used.

DESCRIPTION

This command converts AOCV tables annotated on design objects (previously loaded by **read_ocvm** command) to POCV tables and applies the converted POCV tables onto the same design object.

The AOCV to POCV conversion is based on the following equation.

POCV_coefficient_late = (AOCV_derate - 1) / 3
POCV_coefficient_early = (1 - AOCV_derate) / 3

In the above equation, AOCV_derate refers to the *depth_row* entry in the AOCV table. If the AOCV table has distance information, the worst-case distance is used.

One advantage of using this command is reducing the pessimism in AOCV GBA analysis.

To report the converted POCV tables, use **report_ocvm** command. To remove the converted POCV tables, use **remove_ocvm**

command.

Multicorner-Multimode Support

By default, this command works on the current corner. To specify a different corner, use the **-corners** option.

EXAMPLES

The following example converts the AOCV tables for corner1.

```
prompt> convert_aocv_pocv -corners corner1
```

SEE ALSO

read_ocvm(2)
report_ocvm(2)
remove_ocvm(2)

convert_shape_patterns_to_shapes

Converts shape_patterns to shapes in the current design.

SYNTAX

collection **convert_shape_patterns_to_shapes**
[-verbose]
[-force]
shape_pattern_list

Data Types

shape_pattern_list collection

ARGUMENTS

-verbose

Display additional debugging information.

-force

Ignore the locked status of objects. If this option is not provided and any of the affected objects has locked status, the command will generate a Tcl error and exit.

shape_pattern_list

Specifies a list of shape_patterns to convert and replace with shapes. The list can contain shape_pattern collections specified with the **get_shape_patterns** command.

DESCRIPTION

This command creates new shapes to replace the specified shape_patterns and returns a collection containing the new shapes.

Upon success the specified shape_patterns would be removed and their replacement shapes created by this command are returned.

EXAMPLES

The following example converts a shape_pattern to shapes.

```
prompt> convert_shape_patterns_to_shapes [get_shape_patterns RECT_PATTERN_31_0]
{RECT_31_1829 RECT_31_1830 RECT_31_1831 RECT_31_1832}
```

SEE ALSO

create_shape(2)
get_shape_patterns(2)
get_shapes(2)
remove_shape_patterns(2)
remove_shapes(2)

convert_shapes_to_shape_pattern

Converts shapes to shape patterns in the current design.

SYNTAX

collection **convert_shapes_to_shape_pattern**
[-base_shape *shape*]
[-m_displacement *delta*]
[-n_displacement *delta*]
[-verbose]
[-force]
[-shapes *shapes*]
[-region *geometry*]
[-layers *layers*]
[shape_list]

Data Types

base_shape collection
delta distance:direction | {x-distance:x-direction
y-distance:y-direction}
distance float
direction string
shapes collection
geometry collection
layers collection
shape_list collection

ARGUMENTS

-base_shape

Specifies the shape in the input list that should be used as the base_shape at the first (0,0) position.

-m_displacement

Specifies the uniform offset in first dimension.

-n_displacement

Specifies the uniform offset in second dimension.

-verbose

Display additional debugging information.

-force

Ignores the locked status of objects. If this option is not provided and any of the affected objects has locked status, command will generate a Tcl error and exit.

-shapes *shapes*

Specifies a list of shapes to convert and replace with shape patterns. The list can contain shape collections specified with the **get_shapes** command. The option "**shapes**" is mutually exclusive with "**region**", "**layers**" and "**shape_list**".

-region *geometry*

Specifies a list of regions to get shapes and convert and replace them with shape patterns. The option "**region**" is mutually exclusive with "**shapes**", "**layers**" and "**shape_list**".

-layers *layers*

Specifies a list of layers to get shapes and convert and replace them with shape patterns. The option "**layers**" is mutually exclusive with "**shapes**", "**region**" and "**shape_list**".

shape_list

Specifies a list of shapes to convert and replace with shape patterns. The list can contain shape collections specified with the **get_shapes** command. The option "**shape_list**" is mutually exclusive with "**shapes**", "**region**" and "**layers**".

DESCRIPTION

This command creates new shape patterns to replace the specified shapes and returns a collection containing the new shape patterns.

If the optional option *shape_list* is specified, all input shapes must have the same shape_type, owner net and uniform spacing. In case the order of shapes in input collection is different from the expected positional order of the shapes in the converted shape_pattern, the base_shape must be specified to guide the conversion. In case of non-uniform spacing between the input shapes caused by the absence of shape at any position, the user must provide the expected uniform spacing in both dimensions to guide the conversion. The tool would error violation of these requirements and abort the conversion.

The option "**shapes**" and "**layers**" and "**region**" are mutually exclusive with "**shape_list**" and useful for auto detection of via matrixes.

EXAMPLES

The following example converts all shapes to shape patterns.

```
prompt> convert_shapes_to_shape_pattern -shapes [get_shapes]
{RECT_PATTERN_31_0 RECT_PATTERN_31_1}
```

The following example converts specified shapes to a shape pattern.

```
prompt> convert_shapes_to_shape_pattern [get_shapes \
            -filter {shape_type==rect && layer_name==M1}]
{RECT_PATTERN_31_0}
```

The following example converts shapes to a shape_pattern where the last shape in the input collection is used as the base_shape and the uniform displacements are specified to guide the conversion to shape_pattern with unoccupied (hole) positions.

```
prompt> convert_shapes_to_shape_pattern \
            -base_shape [index_collection [get_shapes] end] \
            -m_displacement {50:north} \
            -n_displacement {20:east} \
            [get_shapes -filter {shape_type==rect && layer_name==M1}]
{RECT_PATTERN_31_0}
```

SEE ALSO

create_shape_pattern(2)
get_shape_patterns(2)
get_shapes(2)
remove_shape_patterns(2)
remove_shapes(2)

convert_via_matrixes_to_vias

Converts via matrixes to vias in the current design.

SYNTAX

collection **convert_via_matrixes_to_vias**
[-verbose]
[-force]
via_matrix_list

Data Types

via_matrix_list collection

ARGUMENTS

-verbose

Display additional debugging information.

-force

Ignore the locked status of objects. If this option is not provided and any of the affected objects has locked status, the command will generate a Tcl error and exit.

via_matrix_list

Specifies a list of via matrixes to convert and replace with vias. The list can contain via matrix collections specified with the **get_via_matrixes** command.

DESCRIPTION

This command creates new vias to replace the specified via matrixes and returns a collection containing the new vias.

Upon success the specified via matrixes would be removed and their replacement vias created by this command are returned.

EXAMPLES

The following example converts a via matrix to vias.

```
prompt> convert_via_matrixes_to_vias [get_via_matrixes VIA_MATRIX_0]
{VIA_S_0 VIA_S_1 VIA_S_2 VIA_S_3 VIA_S_4 VIA_S_5}
```

SEE ALSO

create_via(2)
get_via_matrixes(2)
get_vias(2)
remove_via_matrixes(2)
remove_vias(2)

convert_vias_to_via_matrix

Converts vias to via matrixes in the current design.

SYNTAX

collection **convert_vias_to_via_matrix**
[-verbose]
[-force]
[-vias *vias*]
[-region *geometry*]
[-via_defs *via_defs*]
via_list

Data Types

via_list collection
vias collection
geometry collection
via_defs collection

ARGUMENTS

-verbose

Display additional debugging information.

-force

Ignores the locked status of objects. If this option is not provided and any of the affected objects has locked status, command will generate a Tcl error and exit.

via_list

Specifies a list of vias to convert and replace with a via matrix. The list can contain via collections specified with the **get_vias** command. The option "**via_list**" is mutually exclusive with "**vias**", "**region**" and "**via_defs**".

-vias *vias*

Specifies a list of vias to convert and replace with via matrixes. The list can contain via collections specified with the **get_vias** command. The option "**vias**" is mutually exclusive with "**via_list**", "**region**" and "**via_defs**".

-region *geometry*

Specifies a list of regions to get vias and convert and replace them with via matrixes. The option "**region**" is mutually exclusive with "**via_list**", "**vias**" and "**via_defs**".

-via_defs ***via_defs***

Specifies a list of via_defs to get vias and convert and replace them with via matrixes. The option "**via_defs**" is mutually exclusive with "**via_list**", "**vias**" and "**region**"

DESCRIPTION

This command creates new via matrixes to replace the specified vias and returns a collection containing the new via matrixes. If the optional option *via_list* is specified, the specified vias must be of the same via_def and via_type because the base via of the via matrix can be one of a simple, simple array, or custom via. If the specified vias don't have matching attributes for conversion to base via representation of a via matrix, the command would fail retaining the specified vias.

Upon success the specified vias would be removed and their replacement via matrix created by this command is returned. The option "**vias**" and "**via_defs**" and "**region**" are mutually exclusive with "**via_list**" and useful for auto detection of via matrixes.

EXAMPLES

The following example converts vias to a via matrix with a base simple via.

```
prompt> convert_vias_to_via_matrix [get_vias -filter {via_def_name==VIA23 && via_type==simple_via}]
{VIA_MATRIX_0}
```

The following example converts all vias to via matrixes.

```
prompt> convert_vias_to_via_matrix -vias [get_vias]
{VIA_MATRIX_0 VIA_MATRIX_1}
```

SEE ALSO

create_via_matrix(2)
get_via_matrixes(2)
get_vias(2)
remove_via_matrixes(2)
remove_vias(2)

copy_block

Copies a block to a new block in the same or a different library and view

SYNTAX

collection **copy_block**
[-force]
[-from_block *source_block*]
-to_block *destination_block*

Data Types

source_block string
destination_block string

ARGUMENTS

-force

Forces overwrite of the block specified by -to_block *block_name*. This option is used when the destination is open and modified but not yet saved.

-from_block *source_block*

Specifies the source block name with optional library, label, and view specifications. The *source_block* specification is as follows:

[libName:]blockName[/labelName][.viewName]

or a collection of one block. If the library specification is not present, the current library is used. If label specification is not present, then the default label is used. If the view specification is not present, the design view is used. By default, the **current_block** is used as the source.

-to_block *destination_block*

Specifies the destination block name with optional library and view specifications. The *destination_block* specification is as follows:

[libName:]blockName[/labelName][.viewName]

If the library, block, or view specifications are not present, the values from the source block are used. This option is required.

DESCRIPTION

This command copies a source block to a destination with the same or different library and block. The destination block will have a mode attribute with a value of "a" and the destination block can be further modified without any change in open mode. If you try to copy a block over itself, the command issues an error message. The command fails if the destination library is not present or is opened in read-only mode. The command also fails if the destination block is open and has been modified but not saved and the **-force** option is omitted. The command also fails if you specify an illegal name for the source or destination library, block, or view. If successful, the destination block is returned.

This command returns the block if successful, or O if not. If an illegal name is given for the source or destination block names, a Tcl error is raised.

When no view is explicitly specified for both source and destination then all available views of source block are copied into destination block and the design view of destination block is returned.

A given source block view can't be copied into a different destination block view, like a design view can't be copied to an abstract view. An error is issued while trying to do such operation.

The copied destination block is by default available in memory only. This behavior can be controlled using app option **design.on_disk_operation** where setting the app option to true means copied destination block will be saved to disk.

The copied destination block doesn't become the current block. It needs to be explicitly set.

EXAMPLES

This example copies block *Top* in library *lib* to block *Top2* in same library for subsequent block exploration. The destination library and view are taken from the original block.

```
prompt> copy_block -from_block lib:Top -to_block Top2
{"lib:Top2.design"}
prompt> current_block lib:Top2
```

If library *lib* is already open and is current library then using *-from_block Top* and *current_block Top2* in above example is sufficient.

This example copies a block *Top* in current library into a different existing library MyOtherDesigns for subsequent block exploration.

```
prompt> copy_block -from_block Top -to_block MyOtherDesigns:Top
{"MyOtherDesigns:Top.design"}
prompt> current_block MyOtherDesigns:Top
```

This example copies a block *Top* in current library into a different existing library MyOtherDesigns with a different name *Top2* and saves it to disk.

```
prompt> set_app_options -name design.on_disk_operation -value true
prompt> copy_block -from_block Top -to_block MyOtherDesigns:Top2
Information: Saving block 'MyOtherDesigns:Top2.design'
{"MyOtherDesigns:Top2.design"}
```

SEE ALSO

get_blocks(2)
get_designs(2)
move_block(2)
open_block(2)

open_lib(2)
remove_blocks(2)
reopen_block(2)
save_block(2)

copy_busplan

Copy specified parts of busplan.

SYNTAX

int **copy_busplan**

[-from_start *element*]
[-from_end *elements*]
[-to_start *element*]
[-to_end *elements*]
[-from *bus*]

Data Types

element bus elements
elements collection of bus elements
bus bus

ARGUMENTS

-from_start *element*

Specify the starting bus element to copy from. This is mutually exclusive with -from.

-from_end *elements*

Specify the ending bus elements to copy from. This is mutually exclusive with -from.

-to_start *element*

Specify the starting bus element to copy to. This is mutually exclusive with -from.

-to_end *elements*

Specify the ending bus elements to copy to. This is mutually exclusive with -from.

-from *bus*

Specify the busplan to copy from. This will automatically determine the parts of the bus going through MIBs. It will determine the other MIB instances and the buses going through the other MIB instances. It will then copy the MIB section from the given bus to the other buses.

Note that all the buses must all be MIB port compliant. This means that all the buses must use the same ports in the MIB. For example suppose there are 2 MIBS, B1 and B2, and if there is feedthrough bus element corresponding to the ports B1/data[0..10] and there is a feedthrough bus element corresponding to the ports B2/data[0..10]. Then in this case these 2 MIB bus port sets are

MIB port compliant. On the other hand if the feedthrough bus element is different, eg, corresponding to the ports B2/data[0..9] then that is not MIB port compliant.

DESCRIPTION

Copy busplan virtual pins, virtual registers, and location information. There are 2 mechanisms. First, it can be used to copy sub-sections of buses. Using the 4 options, "-from_start *element*", "-from_end *elements*", "-to_start *element*", "-to_end *elements*" a from sub-section and to sub-section can be defined. Then copy will copy from the from sub-section to the to sub-section.

A second mechanism is using option "-from *bus*" and this is used to ensure busplans inside MIBs are the same.

EXAMPLES

This example shows copying a section of bus bus1 to another bus, bus2. The section of bus1 to copy is starting from bus element bus1:group2 and going to bus1:group3. This section will be copied to the section of bus2 starting from bus element bus2:group3 and going to bus2:group4.

```
prompt> copy_busplan -from_start [filter_collection \
  [get_attribute [get_busplan bus1] elements] name==group2] \
  -from_end [filter_collection \
    [get_attribute [get_busplan bus1] elements] name==group3] \
  -to_start [filter_collection \
    [get_attribute [get_busplan bus2] elements] name==group3] \
  -to_end [filter_collection \
    [get_attribute [get_busplan bus2] elements] name==group4]
1
```

SEE ALSO

create_busplans(2)
set_net_estimation_rule(2)

copy_collection

Duplicates the contents of a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

collection **copy_collection**
collection1

Data Types

collection1 collection

ARGUMENTS

collection1

Specifies the collection to be copied. If an empty string is used for the *collection1* argument, the command returns the empty string (a copy of the empty collection is an empty collection).

DESCRIPTION

This command is no longer needed and is provided only to make old scripts work without modification.

EXAMPLES

The following example from PrimeTime shows the result of copying a collection. Functionally, it is not much different that having multiple references to the same collection, except it is slower.

```
pt_shell> set c1 [get_cells "U1*"]
{"U1", "U10", "U11", "U12"}
pt_shell> set c2 [copy_collection $c1]
{"U1", "U10", "U11", "U12"}
pt_shell> unset c1
pt_shell> query_objects $c2
{"U1", "U10", "U11", "U12"}
```

SEE ALSO

collections(2)

copy_floorplan

Copies the floorplan data from the design view of source design label to the design view of the target design label in the same or a different library.

SYNTAX

collection **copy_floorplan**
[-to_block *top_block*]
-from_block *from_top_block*
[-objects *objects*]
[-include *include_list*]
[-exclude *exclude_list*]
[-hierarchical]
[-force]
[-verbose]

Data Types

top_block	string
from_top_block	string
objects	collection
include_list	list
exclude_list	list

ARGUMENTS

-force

Forces copying of those floorplan objects of source block to the destination block even if they already exist. To check existing floorplan objects, names are matched. In the default mode, matching objects are not copied.

-from_block *from_top_block*

Specifies the source design label name with optional library, block and label specifications. The *from_top_block* specification is as follows:

[libName:]blockName[/labelName]

If the library specification is not present, the current library is used. If label specification is not present, then the default label is used. This is a required option.

-to_block *top_block*

Specifies the destination design label name with optional library and label specifications. The *top_block* specification is as follows:

[libName:]blockName[/labelName]

If the library, block, or label specifications are not present, the values from the source block are used. By default, the **current_block** is used as the destination.

-include ***include_list***

Specifies the types of floorplan data to copied to the output design view. This option cannot be specified together with the **-exclude** or **-objects** options. If this option is not specified, all types are included by default.

The supported types are:

- blockages
- bounds
- cells
- corridors
- die_area
- edit_groups
- io_guides
- macros
- module_boundaries
- pin_guides
- pins
- route_guides
- rows
- tracks
- vias
- scan_chains
- routing_directions
- voltage_areas
- fills
- pg_metal_fills
- routing_rules
- pg_regions
- port
- supernet
- bundle
- topology_plan

Specify **-include macros** to output only macro cells and skip standard cells. Specify **-include cells** to output all cells.

-exclude *exclude_list*

Specifies the types of data to exclude from the output. This option cannot be specified with the **-include** option or **-objects** options. If this option is specified, the command writes all types, except the types specified in *exclude_list*. The allowed data types are the same as for the **-include** option.

-objects *objects*

Specifies the floorplan objects to be copied to the destination design view. Supported object types are:

- "block"
- "design"
- "routing_blockage"
- "shaping_blockage"
- "placement_blockage"
- "bound"
- "cell"
- "routing_corridor"
- "core_area"
- "edit_group"
- "fill_cell"
- "io_guide"
- "io_guide"
- "via"
- "shape"
- "net"
- "pin_guide"
- "pg_region"
- "routing_guide"
- "site_row"
- "site_array"
- "track"
- "voltage_area"
- "routing_rule"
- "port"
- "supernet"

- "bundle"
- "topology_plan"

-verbose

This option enables display of verbose messages.

-hierarchical

This option enables copying of the floorplan objects hierarchically. This option is mutually exclusive with "*-objects*".

DESCRIPTION

This command copies the floorplan data from the design view of source design label to the design view of the target design label with the same or different library and block. The command fails if the destination library is not open or is opened in read-only mode. The command also fails if you specify an illegal name for the source or destination design label or if the design view is not available for a particular design label. If successful, the destination block is returned.

This command returns the block if successful, or O if not. If an illegal name is given for the source or destination block names, a Tcl error is raised.

The operation happens only between design views. If the destination design is not specified, the current block is used as the destination design if the view is design view.

The command only copies the floorplan objects which exist in the source but not in the destination. To find whether a particular floorplan object exists, the name is matched. If the name of a particular floorplan object already exists, the object is skipped and is not copied. This can be overriden by specifying the **-force** option.

EXAMPLES

This example copies floorplan data of design view of design label *Top* to design view of block *Top2*. Only the floorplan objects not present in Top2 are copied.

```
prompt> copy_floorplan -from_block lib:Top/place_opt -to_block lib:Top2/place_opt
{"lib:Top2.design"}
```

This example copies a block Top into a new library MyOtherDesigns. Only the floorplan objects not present in Top2 are copied MyOtherDesigns:TOP\place_opt.

```
prompt> copy_floorplan -from_block lib:Top/place_opt -to_block MyOtherDesigns:TOP\place_opt

{"MyOtherDesigns:Top.design"}
```

SEE ALSO

get_blocks(2)
open_block(2)

open_lib(2)
remove_blocks(2)
reopen_block(2)
copy_block(2)
save_block(2)

copy_lib

Copies one or more design libraries from one location to another.

SYNTAX

collection **copy_lib**
[-force]
[-merge | -no_designs]
[-from_lib *source_name*]
-to_lib *destination_name*

Data Types

source_name collection
destination_name string

ARGUMENTS

-force

Overwrites the destination library when the library is modified but not yet saved. If this option is not specified, the command stops and issues an error message if the library is modified but not yet saved.

-merge

Merges the source library into the destination library. This option is used for incremental update. Normally, the destination library is (re)initialized before the copy begins. This option can be used only if both the source and destination libraries are design libraries. Note that lib-cell libraries cannot be merged, and library attributes, technology, and parasitic_tech data are not merged. If the destination library does not already contain technology or parasitic_tech data, the command copies the technology or tech_parasitic data from the source library to the destination library. Only libraries with compatible technology can be merged, and the command performs a technology compatibility test if both the source and destination libraries contain technology sections. The **-merge** option fails if the source and destination technologies are incompatible.

-no_designs

Copies the source library to the destination library, without copying the designs in the source library. This option is mutually exclusive with the **-merge** option. The command copies the library attributes, ref_lib list, view_switch list, technology, and parasitic_tech from the source library without copying any of the designs. This option is useful when splitting a source library that contains subblocks into a separate destination libraries that each contain a subblock.

-from_lib *source_name*

Specifies the source library. You can specify either a list of library names or a collection of libraries. If you specify a list or collection of multiple libraries, the designs in each library are copied to the destination library. If there are duplicate designs in the source libraries, the design from the first library in the list that contains the duplicate is copied to the destination library. If the **-**

no_designs option is used, the **-from_lib** option must contain a single library. If this option is not specified, the command uses the current library. Use the **current_lib** command to determine the current library.

-to_lib ***destination_name***

Specifies the destination library name.. This can be a simple, relative, or absolute path. If it is a relative or absolute path, the trailing portion of the path after he last '/' becomes the name of the library. It will be an error if the a library, file, or directory of the same name already exists on disk.

DESCRIPTION

This command copies one or more source libraries into a destination library. If technology data and a ref_lib list exists, they are copied as well. Note that the command fails if there are conflicts in technology data among the source libraries or between the source and destination libraries when the **-merge** option is used. When there are conflicts in design data, the data from the first library in the source list is used (or from the destination library if the **-merge** option is used and a conflict exists). Note that conflicts are resolved on the design level, not down to the view level. For example, if a source library has topLib1:Mid.timing and another source library has topLib2:Mid.design, only the first design is copied. The destination target is reinitialized (deleted) before the copy unless the **-merge** option is used. When the source and destination libraries contain a reference library list, the source reference list is appended to the target library and the duplicate entries are removed.

The command returns the library if successful, or 0 otherwise. If an illegal name is given, a Tcl error is raised.

Note that when overriding an existing library (without the **-merge** option), there is no automatic rebinding to the new design or technology data in the library. All original designs in the library are removed, and designs from other libraries that used to bind to those removed designs become unbound. Any other libraries or designs that used to bind to the technology data of the overridden library are also be unbound and are saved without a technology section.

It is also an error to copy (or merge) from (or into) lib-cell library. It is also an error to copy over a library that is opened for read-only, or to copy over a library that has unsaved and modified data without using the **-force** option.

EXAMPLES

The following example copies library lib1 into the library design_lib.

```
prompt> copy_lib -from_lib "lib1" -to_lib design_lib
{"design_lib"}
```

The following example copies the designs from lib1 and lib2 into the library top_lib.

```
prompt> copy_lib -from_lib "lib1 lib2" -to_lib top_lib
{"top_lib"}
```

The following example merges the designs from lib1 and lib2 into the library mid_lib, without reinitializing mid_lib

```
prompt> copy_lib -merge -from_lib "lib1 lib2" -to_lib mid_lib
{"mid_lib"}
```

The following example copies the technology and ref_lib list from lib1 to lib2 without copying the designs.

```
prompt> copy_lib -no_designs -from_lib "lib1" -to_lib lib2
{"lib2"}
```

SEE ALSO

close_lib(2)
create_lib(2)
current_lib(2)
get_libs(2)
move_lib(2)
open_lib(2)
save_lib(2)
search_path(3)
set_ref_libs(2)

copy_module

Copies a module in a design

SYNTAX

collection **copy_module**
[-design *design*]
-from *module*
-to *to_module*]
[-force]

Data Types

design string or collection
module string or collection
to_module string

ARGUMENTS

-design *design*

Specifies the design in which to copy the module. If not specified, then module is copied in the current design.

-from *module*

Specifies source module to be copied, as a string or collection. If specified as a string then it is searched in current design.

-to *to_module*

Specifies destination module name.

-force

Forces overwrite of the module specified by **-to** only if it is not the top module of its design and is not instantiated anywhere.

DESCRIPTION

This command copies a specified module to a destination design provided it doesn't already exist. If it already exists in destination then **-force** can be used to overwrite the module provided it is not instantiated anywhere and is not the top module.

A module is hierarchically copied over to the destination. Thus this command can result in copying over many modules into destination. If the module is copied over to a different library then it is checked that reference library of destination library is a subset

of reference library of source library. Otherwise the command will fail.

SEE ALSO

get_modules(2)
get_designs(2)
move_block(2)
remove_modules(2)

copy_objects

Copies specified objects and creates multiple copies if specified.

SYNTAX

status **copy_objects**
[*object_list*]
[-delta {*x y*}]
[-from {*x y*}]
[-to {*x y*}]
[-rotate_by *angle*]
[-group]
[-x_times *x_times*]
[-y_times *y_times*]
[-x_pitch *x_pitch*]
[-y_pitch *y_pitch*]
[-x_pitch_type gap | delta]
[-y_pitch_type gap | delta]
[-net *net*]

Data Types

object_list collection or selection
x float
y float
angle string
x_times integer
y_times integer
x_pitch float
y_pitch float
net collection

ARGUMENTS

object_list

Specifies a collection or selection set that contains objects to copy. If this option is not specified, the global selection set is used.

-delta {*x y*}

Specifies the distance in the x- and y-directions which to copy the objects from their current location. This option is mutually exclusive with the **-to** option.

-from {*x y*}

Specifies the reference point on the object, or the collection of objects to be copied, for the **-to** option. By default, the command uses the lower-left corner of the bounding box of the given object, or collection of objects, as the reference point. This option must be used together with the **-to** option.

-to {*x y*}

Specifies the new x- and y-location of the reference point for the object or objects. This option is mutually exclusive with the **-delta**.

-rotate_by *angle*

Specifies the rotation angle of the objects.

The valid values are:

0, **90**, **180**, **270**, **CW90**,
CW180, **CW270**, **CCW90**, **CCW180**, **CCW270**,
FLIPX, **FLIPY** **R0**, **R90**, **R180**,
R270, **MX**, **MXR90**, **MY**, **MYR90**

The rotation value definitions are:

CW90 : 90 degrees clockwise
CW180 : 180 degrees
CCW180 : 180 degrees
CW270 : 270 degrees clockwise
CCW90 : 90 degrees counterclockwise
CCW270 : 270 degrees counterclockwise
FLIPX : reflect about the x-axis
FLIPY : reflect about the y-axis

The following values are synonymous:

0, **R0**
90, **CCW90**, **R90**
180, **CCW180**, **R180**
270, **CCW270**, **R270**
FLIPX, **MY**
FLIPY, **MX**

-group

Specifies that the objects should rotate around a common pivot point. If omitted, the rotation is performed around the object centers.

-x_times *x_times*

Specifies the number of copies in x-direction.

-y_times *y_times*

Specifies the number of copies in y-direction.

-x_pitch *x_pitch*

Specifies the delta or gap in x-direction.

-y_pitch *y_pitch*

Specifies the delta or gap in y-direction.

-x_pitch_type gap | **delta**

Specifies the x pitch as delta or gap. Legal values are gap or delta.

-y_pitch_type gap | **delta**

Specifies the y pitch as delta or gap Legal values are gap or delta.

-net ***net***

Specifies the net for the copied object.

DESCRIPTION

This command copies one or more objects and returns a collection of the copies if input *object_list* is a collection of objects. Otherwise, the command adds copies in specified selection set and returns the status.

You can place the copies at a specific location or specify a delta from the location of the original object.

You can create multiple copies of each specified object, and you can specify the pitch between adjacent copies of the same object.

All the attributes of an original object are applied to its copies if possible.

EXAMPLES

The following example copies the selected object and places the copy 100 database units to the right and 200 database units above the original object:

prompt> **copy_objects -from {0 0} -to {100 200}**

The following example creates 8 copies of the selected object placed in a 2 by 4 grid with delta (100,100) and no space between the objects.

prompt> **copy_objects -from {0 0} -to {100 100} -x_times 2 -y_times 4**

The following example uses an alternative syntax to specify the objects.

prompt> **copy_objects [get_selection] -from {0 0} -to {100 100} -x_times 2 -y_times 4**

SEE ALSO

change_selection(2)
get_edit_setting(2)
get_selection(2)
get_snap_setting(2)
move_objects(2)
rotate_objects(2)
set_edit_setting(2)
set_snap_setting(2)

snap_objects(2)

copy_relative_placement

Copies the relative placement of macros.

SYNTAX

collection **copy_relative_placement**
-from *edit_group*
[-to *edit_group_list*]
[-to_cells *cell_list*]

Data Types

edit_group string
edit_group_list list
cell_list list

ARGUMENTS

-from *edit_group*

Specifies the edit group from which to copy the relative placement.

-to *edit_group_list*

Specifies a list of edit groups; relative placement will be copied to each edit group in the list.

-to_cells *cell_list*

Specifies a list of cells to which to copy the relative placement. The tool creates an edit group for these cells.

DESCRIPTION

This command copies macro relative locations from one edit group to another edit groups or a collection of macros. At least one of **-to** and **-to_cells** must be provided. Use this command to arrange the corresponding macros from similar modules in the same way. The command returns the collection of edit groups passed as **-to** argument and/or the edit group created for **-to_cells** argument.

EXAMPLES

The following example copies relative placement from edit group E1 to edit groups E2 and E3.

prompt> **copy_relative_placement -from [get_edit_groups E1] -to [get_edit_groups {E2 E3}]**

The following example copies relative placement from edit group E to the selected macros.

prompt> **copy_relative_placement -from [get_edit_groups E] -to_cells [get_selection]**

SEE ALSO

create_edit_group(2)

copy_to_layer

Create shapes on a layer corresponding to a geometric region.

SYNTAX

collection **copy_to_layer**
-layer *layer*
-geo_masks *geo_masks*
[-shape_use *use*]
[-net *net*]

Data Types

layer collection
geo_masks collection
use string
net collection

ARGUMENTS

-layer *layer*

Specifies the layer on which shapes should be created. There must be exactly one layer in this input collection.

This is a required option.

-geo_masks *geo_masks*

Specifies the objects that define the geometric region to be mapped to shapes on the specified layer. If the *shape_use* option is detail_route, one shape will be created on the layer for each contiguous poly-rectangular region contained in the collection of geo_masks. If the *shape_use* option is stripe, then one shape will be created on the layer for each rectangle in the rectangular decomposition (fracture) of the collection of geo_masks.

This is a required option.

-shape_use *use*

Specifies the usage of the shapes created by the command.

Valid values are

- **detail_route** (the default)
  Shapes are used for detail routed signal nets. They will be of type rect or polygon.

- **stripe**
  Shapes that belong to a power strap. They will be of type path.
- **pg_augmentation**
  Shapes for PG augmentation. They will be of type path.

-net ***net***

Specifies the net that will own the resulting shapes.

DESCRIPTION

This command creates shapes on the specified layer corresponding to a specified geometric region.

When the **shape_use** option is set to detail_route or is unspecified, then a shape will be created for each contiguous poly-rectangular region described by *geo_masks*. For each such region, if it is rectangular and manhattan-angled then a rectangle will be created, otherwise a polygon will be created.

When the **shape_use** option is set to stripe or pg_augmentation, then a path shape will be created for each rectangle in the rectangular decomposition (fracture) of the poly-rectangular region described by *geo_masks*. The orientation of the paths will match the routing_direction of the specified layer, or will be horizontal if the routing_direction is unknown.

The new shapes created are returned as a collection.

EXAMPLES

The following example creates a rectangle on a layer:

```
prompt> set pr [create_poly_rect -boundary {{0 0} {200 100}}]
{{{0.0000 0.0000} {200.0000 100.0000}}}
prompt> set new_shapes [copy_to_layer -layer [get_layers M1] \
  -geo_masks [create_geo_mask -objects $pr]]
{RECT_10_0}
```

SEE ALSO

create_poly_rect(2)
create_geo_mask(2)
compute_area(2)
compute_polygons(2)
resize_polygons(2)
split_polygons(2)
get_shapes(2)
transform_polygons(2)

cputime

Retrieves the overall user time associated with the current process.

SYNTAX

float **get_cputime**
  [-all]
  [-format]

string *format*

ARGUMENTS

all

If specified then it gives cpu time for this process and it's children.

format

This argument takes a *printf* style formatting string for a floating point number.

DESCRIPTION

The *get_cputime* command returns the accumulated user time, in seconds, for the current process. If the *format* string is omitted, an integer number of seconds is returned. If the format string is specified, a floating point number is returned. The number includes fractions of a second elapsed and is formatted in accordance with given specifier. Refer to the Unix man page for *printf* for a detailed description of how to format a floating point argument.

```
prompt> get_cputime "%g"
3.74
```

SEE ALSO

get_mem(2)

create_3d_mirror_bumps

Copies and places bumps and bump clusters or bond pads from the source die to the target die.

SYNTAX

status **create_3d_mirror_bumps**
-from *source_chip_name*
-to *target_chip_name*
[-ref_cell *lib_cell*]
[-prefix *prefix_name*]
[-bumps *bump_list*]
[-cluster_bump_mapping *pattern_cell_pair_list*]
[-bond_pad]
[-pins *pin_lis*]
[-force]

Data Types

source_chip_name string
target_chip_name string
lib_cell string
prefix_name string
bump_list list
pattern_cell_pair_list list
pin_lis list

ARGUMENTS

-from *source_chip_name*

Specifies the source chip as a master die. This option is required.

-to *target_chip_name*

Specifies the target chip as a slave die. This option is required.

-ref_cell *lib_cell*

Specifies the library cell. This option is required for bump and cannot be specified with the **-bond_pad** option.

-prefix *prefix_name*

Specifies the prefix of the target mirrored objects.

-bumps *bump_list*

Specifies the bump list in source chip to be mirrored. By default, all bumps are mirrored. This option cannot be specified with the **-bond_pad** option.

-cluster_bump_mapping ***pattern_cell_pair_list***

Specifies the bump cluster pattern cells in source chip and target chip. The pattern cell pair list is specified as {{source_pattern_cell target_pattern_cell} {} ...}. This option cannot be specified with the **-bond_pad** option.

-bond_pad

Mirrors bond pads from source chip to target chip.

-pins ***pin_list***

If source chip is a hard macro, this option can be used to specify the pin list in the source chip to be mirrored. By default, all pins will be mirrored if the source chip is a hard macro.

-force

Specifies whether to mirror the objects if there are overlaps.

DESCRIPTION

This command copies and places bumps and bump clusters or bond pads from the source die to the target die when the dies are stacked vertically.

EXAMPLES

The following example copies bump cells from chip "Logic" to chip "Interposer" with the library cell "UBUMP", using the prefix "synopsys".

```
prompt> create_3d_mirror_bumps -from Logic -to Interposer \
  -ref_cell UBUMP -prefix synopsys
```

The following example copies bump cluster which has pattern cell "BUMP8" from chip "Logic" to chip "Interposer" with pattern cell "BUMP4".

```
prompt> create_3d_mirror_bumps -from Logic -to Interposer \
  -cluster_bump_mapping {{BUMP8 BUMP4}}
```

SEE ALSO

set_cell_location(2)

create_3d_top_design

Creates a 3dic virtual top design.

SYNTAX

collection **create_3d_top_design**
[-force]
-max_dimensions *max_dimensions_list*
[-package_type *package_type*]
top_design_name

Data Types

max_dimensions_list list
package_type string
top_design_name string

ARGUMENTS

-force

Overwrites the block if one with the same name already exists in the current library.

-max_dimensions *max_dimensions_list*

Specifies the maximum allowed width and height of the package. This can be used to check if all 3DIC dies and the silicon interposer (if any) can fit into the package. The Z dimension is unlimited.

-package_type *package_type*

Specifies the type of the package substrate. The value can be one of: **fan_out**, **hybrid**, **interposer**,**stacked**, **unspecified**. The default is **unspecified**.

top_design_name

Specifies the top design name.

DESCRIPTION

The **create_3d_top_design** command creates a new block in the current library to be used as the top-level design. There is no manufacturing tape-out associated with this design. The design type of the new block is 3DIC.

EXAMPLES

The following example creates a new block named top in the current library created by **create_lib**.

```
prompt> create_lib top.exp
{top.exp}
prompt> create_3d_top_design -force -max_dimensions {15000 10000} -package_type interposer top
Information: Creating block 'top.design' in library 'top.exp'. (DES-013)
{top.exp:top.design}
prompt> get_attribute [current_block] design_type
3dic
```

SEE ALSO

create_die_block(2)
create_bump_block(2)
create_die(2)
create_lib(2)

create_3d_virtual_blocks

Creates virtual interface blocks.

SYNTAX

collection **create_3d_virtual_blocks**
[-design *design*]
[-blocks { { *die1*[:*num_layers*] *die2*[:*num_layers*] }... }]
[-layers *num_layers*]
[-in_netlist]
[-die_separation *separation_value*]

Data Types

design collection
die1 string
num_layers integer
die2 string
separation_value float

ARGUMENTS

-design *design*

Specifies the design in which to create the virtual interface blocks. If no design is specified, the virtual interface blocks are created in the current design.

-blocks { { *die1*:*num_layers die2*:*num_layers* }... }

Specifies the die cells between which to create the virtual interface block. One or more pairs of die cell names can be specified. The number of routing layers to copy from each die can be specified after each name, which overrides the value specified by the *-layers* option.

-layers *num_layers*

Specifies the number of routing layers to copy from each die. If not specified, three layers are copied from each die. This can be overridden on a per-die basis in the *-blocks* specification.

-in_netlist

Connects the virtual interface blocks to the top netlist between the dies. If not specified, the virtual interface blocks are not connected to the top netlist.

-die_separation *separation_value*

Specifies the die separation distance used for micro bumps in the virtual interface blocks.

DESCRIPTION

This command creates virtual interface blocks between the specified dies or interposer cells. If no dies or interposer cells are specified, this command creates virtual interface blocks between all die and interposers in design which have their z-offset values set.

EXAMPLES

The following example creates virtual interface blocks between all dies and interpose in the design based on their z-offset values. The virtual interface blocks are connected to the top netlist.

```
prompt> create_3d_virtual_blocks -in_netlist
```

The following example creates a virtual interface block between two dies containing four routing layers from each die.

```
prompt> create_3d_virtual_blocks -blocks {die1 die2} -layers 4
```

The following example creates a virtual interface block between two dies containing two routing layers from the first die and three routing layers from the second die.

```
prompt> create_3d_virtual_blocks -blocks {die1:2 die2:3}
```

SEE ALSO

remove_3d_virtual_blocks(2)

create_abstract

Creates abstract view of designs.

SYNTAX

status **create_abstract**
[-placement]
[-estimate_timing]
[-timing_level none | boundary | compact | full_interface]
[-host_options host_option_name]
[-work_dir path]
[-blocks *block_designs*]
[-all_blocks]
[-force_recreate]
[-read_only]
[-include_objects {*include_objects_collection*}]
[-target_use implementation | planning]
[-preserve_block_instances true | false]

Data Types

block_designs list
include_objects_collection collection of ports, pins and nets

ARGUMENTS

-placement

Creates the abstract view with no timing information. When this option is specified, the tool will not trace timing paths to the ports of the design. Instead, the tool examines the logical hierarchy of the design, and creates an abstract view with the necessary netlist data for top-level global placement. If the design has timing constraints loaded, constrained objects will be included in the placement abstract. A placement abstract is always editable. So this option overrides *-read_only*. By default, the generated abstract view contains timing information.

-estimate_timing

Run the **estimate_timing** command on interface logic and incorporate the estimated timing result in the abstract view.

-timing_level none | **boundary** | **compact** | **full_interface**

Specifies the amount of timing detail to include in the abstract. Abstracts with less detail require less memory and CPU.

You can select one of three timing levels:

- **none** : The abstract excludes timing information at the top level. This is the default abstract type when **-placement** is

specified.

Use this abstract for pure physical operations (all design planning steps prior to **estimate_timing**, doing some purely physical operations at top-level).

The value 'none' is deprecated and will be removed in a future release.

- **boundary** : The abstract includes minimal netlist ensuring connectivity of block ports to the first level of logic. Pure feedthrough, clock combinational feedthrough and internal clock path driving output port are retained

  Use this abstract for better transition fixing of the top level segment of boundary nets when abstract is linked to top.

- **compact** : The abstract includes timing information for all critical setup and hold paths at the boundary of the original design. When the abstract is instantiated at the top level, you can use the abstract to show the best and worst timing through any input port or output port of the abstracted design. These abstracts have a very high level of compression when compared to the full design, and have significant runtime and memory benefits. This is the default when **-placement** is not specified.

  When you build a "compact" timing abstract, it is necessary to evaluate timing paths in the design that is being abstracted. Because of this, you should have the following SDC constraints defined in your block, prior to creating the "compact" abstract:

  - All clocks and all generated clocks
  - All set_case_analysis and set_disable_timing
  - All exception constraints (**set_false_path**, **set_multicycle_path**) that affect timing paths that enter or leave the block.

  Use this abstract for all top-level implementation flows.

- **full_interface** : The abstract includes timing information for almost all paths at the boundary of the original design. Because all boundary paths are represented, it is not necessary to prepare a full SDC file for the original design before creating the abstract. However, you should have the following SDC constraints defined in your block before creating the "full_interface" abstract:

  - All clocks and all generated clocks

  "Full_interface" abstracts contain nearly all of the boundary logic in the original design; some boundary logic is filtered out to reduce the abstract size. In particular, logic connected only to high-fanout networks is filtered. Usually, the filtered logic includes clock networks, scan enable, and reset. For these signals, only the essential part of the logic is kept. You can control the filtering of high-fanout networks by setting the "abstract.high_fanout_limit" app option.

  Use this abstract when your block SDC is not ready.

-host_options host_option_name

Specifies that the given host option should be used for distributed processing. If not specified, the tool will look for the global host option and use the global host option if it has distributed processing settings.

When **-blocks** or **-all_blocks** is specified, the tool might need to create abstract view for more than one block. Some of the blocks can run at the same time, depending on the hierarchy nesting.

-work_dir *path*

Specifies the directory to put all generated scripts and log files. This option is only used when **-blocks** or **-all_blocks** is specified. When **-blocks** or **-all_blocks** is specified, the tool need to create abstract view for one or more child blocks of the current block. Scripts and log files are generated for abstract view creation at the child blocks' level. The tool puts all the scripts and log files under a directory on disk.

If this option is not specified, the *work_dir* setting for the specified host option would be used. If no **-host_options** is specified, the *work_dir* of the global host option would be used. If there is no global host option and no **-host_options** is specified, the default is *./work_dir*.

-blocks *block_designs*

Specifies the list of child block designs to create abstract view. The parent design will switch to use the abstract view as reference for the instantiations of the corresponding designs, if they are not referencing to abstract view already. This option is mutually exclusive with **-all_blocks**. If no **-blocks** or **-all_blocks** is specified, the abstract view for the current design is created.

In a multi-level physical hierarchy design, only one level of physical hierarchy can be abstract view.

-all_blocks

Specifies that abstract view is to be created for all the child blocks in the current design, given that the block is the lowest in the physical hierarchy tree that is allowed to have abstract view. The parent design will switch to use the abstract view as reference for instantiations of the corresponding designs, if they are not referencing to abstract view already. This option is mutually exclusive with **-blocks**. If the **-blocks** or **-all_blocks** option is not specified, the command creates an abstract view for the current design.

In a multi-level physical hierarchy design, the tool will create abstract for the lowest level of block.

-force_recreate

This option only works with **-blocks** or **-all_blocks**. Without **-blocks** and **-all_blocks**, the command will always re-create the abstract view of the current block.

When **-blocks** or **-all_blocks** is specified, the tool would first search the physical hierarchy tree looking for the blocks to create abstract view. After the tool finds a certain block that should create abstract view for, it checks whether the block already has an abstract view and whether this abstract view is of the same type the user wants to create now. By default, the tool would not attempt to re-create the abstract view if the existing one is of the same type.

If the **-force_recreate** option is specified, the tool would do the proper view merge, delete the existing abstract view, and create a new abstract view, even if the existing one is of the correct type. This can be useful when the user wants to make sure that the blocks use the up-to-date abstract views.

-read_only

Specifies that the abstract view being created is read-only. By default, when an abstract view is created, the design view of the block is locked and becomes read-only. By specifying this option, you are asking the tool to make the abstract view read-only and keep the editability of the design view.

This option has no effect when **-placement** is specified. The two options are mutually exclusive.

-include_objects {*include_objects_collection*}

Specifies that some objects need to be retained in the abstract. These objects can be ports, leaf pins, hierarchical pins or nets. Specifying a pin (leaf or hierarchical) will cause the cell of the pin to be included in the abstract. Specifying a net *n* will cause the pins on the same flat net as *n* that are also specified in the *include_objects_collection* to be connected in the abstract. At least one leaf-level driver and one leaf-level load on the flat net containing *n* need to be specified for the connection to occur.

This option can be used for both timing and placement abstracts. This option cannot be used together with options **-blocks**, **-all_blocks** and **-estimate_timing**.

-target_use implementation | planning

Specifies the stage in which the created abstract is targeted to be used. Based on the target stage, tool will apply appropriate internal settings.

You can select one of the two stages:

- **implementation** : The abstract is targeted to be used in top-level implementation.

  Abstract creation is allowed in any level of physical hierarchy.

- **planning** : The abstract is targeted to be used in design planning.

  Abstract creation by default is allowed only on blocks which does not have any descendant block that can be abstract view. This can be overridden by setting the **abstract.allow_all_level_abstract** application option to true.

-preserve_block_instances true | **false**

This option is relevant to designs with nested physical hierarchies. This option determines whether lower levels of physical hierarchies are preserved or whether the hierarchy is made part of the current hierarchy. If this option is set to false, tool will create a flattened abstract. The default value of the option is true.

DESCRIPTION

This command creates an abstract view of the current block. If the block already has an abstract view, this command removes the existing abstract and generates a new abstract view. All leaf cells of the block needs to be mapped prior to abstract creation.

This command also saves the current block to disk.

Abstract view contains all the information that is needed for design planning. For top-level implementation, in addition to abstract, create a frame view using the **create_frame** command.

By default, the abstract view is editable and the design view is locked and becomes read-only. You can open and modify the design view, but *save_block* or *save_lib* would not be able to save the modified design view to disk. If you want to save the modified design view, you have a choice. You can either call *remove_abstract* to remove the abstract view and release the write lock on the design view, or call *create_abstract* to remove the old abstract view, save the modified design view to disk and create a new abstract view.

Design Flow for Abstracts

By default, an editable abstract view is created. At the same time, the design view of the block is locked and becomes read-only.

After the abstract view of the block is created, you can change the instantiation of the block in a parent block to reference to the abstract view using the *change_abstract* command.

If the abstract view is editable and instantiated in a parent block, you can make changes to it in the parent block context. Changes in the abstract view are brought back into the full netlist design view through merge. Merge happens automatically when the design view is loaded. You can also call merge explicitly using the command *merge_abstract*.

The merge operation requires that the full netlist design view remain unchanged between the time that the abstract is created with the **create_abstract** command and the abstract is merged back to the design view.

Resuming Editing of the Full Netlist

When an editable abstract view is created. At the same time, the design view of the block is locked and becomes read-only. The read-only design view can still be opened and modified in memory. But if you want to commit the in-memory modifications to disk, you won't be able to do it with *save_block* or *save_lib* command. There are three ways to save the in-memory modification of an abstract-locked design view to disk:

1. You can save it to another design using *save_block -as xxx*.

2. If you no longer need to use the abstract view, you can call *remove_abstract* to delete the abstract view and release the lock on the design view. Then, you will be able to use *save_block* or *save_lib* to save the design view.

3. If you want to get an updated abstract view according to the latest design view in-memory, you can call *create_abstract* command. The *create_abstract* command will delete the existing abstract view, release the lock on the design view, save the design view to disk, create the new abstract view, and lock the design view again if no *-read_only* is specified.

Summary for timing abstract creation

When a timing abstract is created, the tool by default retains

- Data and clock paths needed for analysis and optimization at top-level

  In addition, some cells are retained because they are used in timing constraints or exceptions. The log will contain a summary reporting how many cells were kept because of exceptions. The summary also reports how many pins in the design were affected by each type of exception or constraint. Specifically, the report includes:

  - **User size only** : Number of pins on cells that are size only (set by the command set_size_only)
  - **Exception** : Number of pins referred to by timing exceptions like set_false_path, set_multicycle_path, set_max_delay etc.
  - **Clock** : Number of pins that are a clock source, a generated clock source or a master pin for a generated clock.
  - **Clock Gating** : Number of pins that belong to clock gating cells.
  - **Constraint** : Number of pins referred to in constraints like set_input_delay, set_output_delay, set_max_time_borrow etc.
  - **Case analysis** : Number of pins with case analysis values (defined with set_case_analysis command).
  - **Cell mode** : Number of pins that have cell mode constraints (defined with set_cell_mode command).
  - **Disable timing** : Number of pins that have disable timing exception (defined using the set_disable_timing command).
  - **CTS exception** : Number of pins that have CTS exceptions on them, for example set_clock_balance_points.
  - **PVT** : Number of pins that have PVT exceptions, like set_operating_conditions, set_voltage, set_temperature etc.
  - **Latency** : Number of pins that have a clock latency set (this affects only pins that are not on sequential cells).
  - **Clock sense** : Number of pins that have a set_clock_sense set.

Multi-mode and multi-corner support

During create_abstract, the tool

- identifies and retains interface logic across all active modes
- Retains pins with constraints across all modes (active and inactive)
- Retains parasitics for all corners (active and inactive)

  Ensure that all the scenarios used at top-level are defined and active while creating abstract.

Multivoltage support

During create_abstract, the tool

- stores the UPF relevant to the retained interface logic. This is promoted to the top-level during linking.

Multi-level physical hierarchy design

- Creating abstract for using in Design Planning Flow

  In a multi-level physical hierarchy design, only one level of physical hierarchy can be an abstract view. By default, it's the

lowest level of blocks.

- Creating abstract for Top-level Place and Route

  Nested abstracts are supported for top-level place and route. Set the abstract.allow_all_level_abstract application option.

Creating abstracts with power annotation

- Set the abstract.annotate_power application option to create abstracts with power information. This enables you to perform power analysis of the complete design while at top-level

Creating abstracts for Signal EM Analysis

- Set the abstract.enable_signal_em_analysis application option to true before running **create_abstract**. This enables you to signal EM analysis inside abstracts while at top-level

EXAMPLES

The following example creates an abstract view, with timing information, for the current design.

```
prompt> create_abstract
```

The following example creates an abstract view that is suitable for top-level global placement.

```
prompt> create_abstract -placement
```

The following example creates an abstract view that is suitable for top-level optimization.

```
prompt> create_abstract
```

The following example runs **estimate_timing** on the interface logic of the current design, then creates an abstract view that includes the result of the estimated (virtually optimized) timing.

```
prompt> create_abstract -estimate_timing
```

The following example calls *create_abstract* from the top level, finds all the child blocks that are abstract view or can be an abstract view, and create the up-to-date abstract view including merging any changes from existing abstract view to the design view, also change the instantiations in parent blocks to reference to the block's abstract view.

```
prompt> create_abstract -all_blocks -force_recreate
```

In the following example we assume that the design contains a net n1 that connects a driver D1 to loads L1, L2, L3, L4. Using the sequence of commands shown, will ensure that the abstract contains the cells of the pins D1 and L3 and that D1 and L3 are connected by a net in the abstract.

```
prompt> set c_pins [get_pins {D1 L3}]
prompt> set c_nets [get_nets {n1}]
prompt> set cAll [add_to_collection $c_pins $c_nets]
prompt> create_abstract -include_objects $cAl
```

The following example creates a flattened abstract if a block has nested blocks in it. If the block does not have any nested blocks, it simply creates abstract for the block.

```
prompt> create_abstract -preserve_block_instances false
```

SEE ALSO

change_abstract(2)
create_frame(2)
estimate_timing(2)
get_abstract_type(2)
merge_abstract(2)
remove_abstract(2)
report_abstracts(2)
abstract.allow_all_level_abstract(3)
abstract.annotate_power(3)
abstract.enable_signal_em_analysis(3)
abstract.high_fanout_limit(3)
abstract.latch_levels(3)
plan.flow.design_view_only(3)

create_abut_rules

Creates spacing rules or keepout margins for cells in a design.

SYNTAX

status **create_abut_rules**
[-soft_keepout]
[-hard_keepout]
[-keepout_width *width*]
[-number_of_references *number*]
[-use_lib]
[-output] *filename*

Data Types

width float
number int
filename string

ARGUMENTS

-soft_keepout

Uses soft keepout margins instead of spacing rules.

-hard_keepout

Uses hard keepout margins instead of spacing rules.

-keepout_width *width*

Specifies the soft keepout margin width to use, in microns. This option should only be used if the *-soft_keepout* or *-hard_keepout* option is used. The default value is the site width.

-number_of_references *number*

Specifies the number of top pin_density values, from which to obtain references to apply spacing rules or keepout margins. The default is 6.

-use_lib

Specifies that library reference cells are used as a source for top pin_density lib cells, instead of reference cells found in the design.

-output *filename*

Specifies the output command filename.

DESCRIPTION

The **create_abut_rules** command helps to improve routability of a design by spacing cells further apart with the use of keepout_margins or by flipping cells. Before running this command, run the **check_routes** command to ensure that the DRC error information is up-to-date.

The command **create_abut_rules** creates either spacing rules (default) or soft/hard keepout margins on reference cells with the top six (default) largest pin_density values. These reference cells are found in the design. If the *-use_lib* option is specified, then the libraries are used to derive the reference cells. An output file is written out.

EXAMPLES

The following example runs the **create_abut_rules** command using the top 4 pin density values to derive reference cells found in the design to apply spacing rules to.

```
prompt> create_abut_rules -number_of_references 4 -output abut_4.tcl
```

The following example creates a soft keepout margin of width 2 microns using the libraries as reference cells to apply soft keepout margins to.

```
prompt> create_abut_rules -number_of_references 5 -output soft_abut_5.tcl \
  -soft_keepout -keepout_width 2
```

SEE ALSO

check_routes(2)
create_keepout_margin(2)
set_placement_spacing_label(2)
set_placement_spacing_rule(2)

create_backend_tcd_cells

Inserts back-end testkey critical dimension cells (TCDs) into the design.

SYNTAX

status **create_backend_tcd_cells**
-lib_cells *layer_lib_cells*
-window_size *{width height}*
[-tcd_spacing *{dist_layer1_layer2}*]
[-other_cell_spacing *{dist_layer}*]
[-icovl_spacing *{dist_layer}*]
[-stack_with_macro *layer_list*]
[-stack_with_frontend *layer_list*]
[-stack_with_backend *{layer_list}*]
[-avoid_route_guide *{layer_list}*]
[-density *{density_layer}*]
[-orientation *{orient_layer}*]
[-check_only]
[-include_small_windows]
[-bbox *{{x1 y1} {x2 y2}}*]
[-snap_to_fin_grid]
[-place_at_window_center]
[-combined_tcd_cells]

Data Types

layer_lib_cells string
width float
height float
layer_libcells string
dist_layer1_layer2 string
dist_layer string
layer_list list
density_layer string
orient_layer string

ARGUMENTS

-lib_cells *layer_lib_cells*

Specifies the metal layer and corresponding back-end TCD library cells. For example, { {M1 cell1 cell2} {M2 cell3 cell4} } specifies back-end TCD library cells cell1 and cell2 are for layer M1; library cells cell3 and cell4 are for layer M2.

-window_size ***{width height}***

Specifies width and height of insertion window size.

-tcd_spacing ***{ dist_layer1_layer2}***

Specifies the minimum distance between two layers of back-end TCD cells. For example, {{10 M1 M4} {20 M2 M3} } specifies that the spacing between M1 and M4 TCD cells is 10um, and the spacing between M2 and M3 TCD cells is 20um. If no spacing value is specified, the default spacing is 0um and TCD cells can abut but cannot overlap. The command issues an error message if two back-end TCD layers have a spacing rule specified with **-tcd_spacing** and also have an overlap specified with **-stack_with_backend**.

-other_cell_spacing ***{dist_layer}***

Specifies the minimum distance between back-end TCD cells with non back-end TCDs and ICOVL cells. This rule does not apply to cells specified with the **-stack_with_macro** and **-stack_with_frontend** options. For example, { { 10 M1} {10 M3 } { 20 M2 } } specifies that the minimum spacing between a M1 back-end TCD and a non back-end TCD cell is 10um, and the minimum spacing between a M3 back-end TCD and a non back-end TCD cell is also 10um, and the minimum spacing between a M2 back-end TCD and a non back-end TCD cell is 20um. By default, the default spacing is 0um and cells can abut but cannot overlap.

-icovl_spacing ***{ dist_layer}***

Specifies the spacing between back-end TCDs and ICOVL cells. For example, { { 10 M1} {10 M4} { 20 M2} {20 M3} } specifies that the spacing between ICOVL cell and M1, M4 layer back-end TCDs is 10um, and the spacing between ICOVL cells and M2 and M3 layer back-end TCDs is 20um. By default, the default spacing is 0um and cells can abut but cannot overlap.

-stack_with_macro ***layer_list***

Specifies a list of back-end TCD metal layers that are allowed to stack with macros. For example, {M1 M5} specifies that M1 layer back-end TCDs can stack with macros without M1 layer metal objects, and M5 back-end TCDs can stack with macros without M5 layer metal objects. Other metal layer back-end TCDs are not allowed to stack with any macros. Note that the command will only check if a macro containing any top-level objects on the specified metal layers. It will not open and check cells inside a macro.

-stack_with_frontend ***layers***

Specifies a list of back-end TCD metal layers that can overlap with frontend TCDs. For example, { M1 M3 } specifies that M1 and M3 layer backend TCDs can overlap with frontend TCDs.

-stack_with_backend ***{layer_list}***

Specifies which layers of back-end TCDs can overlap. For example, { { M1 M2} { M3 M4 } } specifies that back-end TCDs of M1 and M2 layers can overlap, and back-end TCDs of M3 and M4 layers can overlap.

-avoid_route_guide ***{layer_list}***

Specifies which layer of back-end TCDs must avoid route guides and routing blockages on another layer. For example, { { M1 M3 } { M2 M4 } } specifies that M1 layer back-end TCDs should avoid route guides and routing blockages of layer M3; and M2 layer back-end TCDs should avoid route guides and routing blockages for layer M4.

-density ***{density_layer}***

Specifies the target percentage of windows that contain a back-end TCD cell. For example, { {80 M1} {80 M3} { 50 M2} } specifies that the target density for M1 layer and M3 layer back-end TCDs is 80%, and the target density for M2 layer back-end TCDs is 50%. The command tries to evenly distribute the TCDs. For example, if the target density is 50%, it tries to insert one back-end TCD for every other window. By default, the density is 100% and each window should have one back-end TCD.

-orientation ***{orient_layer}***

Specifies the back-end TCD orientation. Valid orientation values are R0, R90, R180, R270, MX, MXR90, MY, MYR90. For example, { {R0 M1} {R0 M2} { MX M3} {MX M4} } specifies that M1 and M2 layer back-end TCDs have an orientation of R0, and M3 and M4 layer back-end TCDs have an orientation of MX. If the specified orientation is not one of TCD cell's allowable

orientation (attribute allowable_orientation), the command issues a warning message. By default, the orientation is N.

-check_only

Checks the current design for back-end TCD violations based on rules specified in other options. An error message is issued for each violation.

-include_small_windows

Includes small windows at the last row/column for TCD insertion. By default, these small windows are excluded.

-bbox ***{{x1 y1} {x2 y2}}***

Specifies an insertion area. This area is divided into the specified window size for TCD insertion. By default, the core area is the insertion area.

-snap_to_fin_grid

Places the lower-left corner of the TCD cell to align with the FinFET grid. By default, alignment is not required.

-place_at_window_center

Changes the placement preference of TCD to be close to window center when specified to:

1. Place TCD in center of window

2. Stack with cells specified in -stack_with_* options

3. Place TCD in the channel between blocks or IPs (if blocks are not abut)

4. Place TCD at blocks or IP corner

By default, TCD cells are not placed at the center of window.

-combined_tcd_cells

Treat the cell is a combined front-end and back-end TCD cell. Placement blockages will be generated when combined TCD cells are placed.

DESCRIPTION

This command inserts back-end testkey critical dimension cells (back-end TCDs), which are used to increase design for manufacturability (DFM) and yield.

Back-end TCDs are inserted after power planning and contain only one metal layer. You can stack back-end TCDs over front-end TCDs to save placement area.

TCD width and height are not even multiples of unit tile width and height. They are placed as are regular macros.

Back-end TCDs may stack with front-end TCDs. back-end TCDs with metal layer M cannot place over layer M routing. If a front-end TCD is on layer M routing, then a back-end TCD with metal layer M cannot stack with this front-end TCD.

Back-end TCD insertion should prevent core cell area penalty by honoring the following insertion priority:

- Place the cell at the center of the window if **-place_at_window_center** option is specified
- Stack the cell with cells specified in stack_with_* option
- Place the cell in the channel between blocks or IPs (if blocks are not abut)

- Place the cell in blocks or IP corner
- Place the cell in the core area

EXAMPLES

The following example inserts the back-end TCD cells DM1 and DM2 into windows of width 1000 microns and height of 1200 microns. DM1 has its metal layer on M1. DM2 has its metal layer on M2. Spacing between DM1 is 10 microns. Spacing between DM2 is 20 microns. Spacing between DM1 and other cells is 2 microns. Spacing between DM2 and other cells is 5 microns. DM1 can stack with front-end TCD cells.

```
prompt> create_backend_tcd_cells \
  -lib_cells {{ M1 DM1} {M2 DM2}} -window_size {1000 1200} \
  -tcd_spacing { { 10 M1} { 20 M2 } } \
  -other_cell_spacing { { 2 M1 } { 5 M2 } \
  -stack_with_frontend {M1}
```

SEE ALSO

create_frontend_tcd_cells(2)
check_tcd_cells(2)

create_blackbox

Creates a black box design and updates the reference for the specified cell.

SYNTAX

status **create_blackbox**
[-new_name *bb_name*]
[-type physical | macro]
[-library *lib_name*]
[-target_boundary_area *area*]
[-boundary *boundary*]
cell

Data Types

bb_name string
lib_name string
area float
boundary list
cell list

ARGUMENTS

-new_name *bb_name*

Specifies the name of the new design. This is an optional argument. This option is only used when creating blackbox on a cell that has unresolved reference. It is ignored if the command is issued on a successfully bound logical hierarchy cell. By default, the command creates a design with the same name as the current reference of the specified cell.

-type physical | macro

Specifies the type of black box to create. Physical black boxes are physical blocks and follow the Physical Partition Design Planning Flow. The logical or macro type black boxes are created for unbound instances and empty hierarchies. After creating the black box, it is treated as any other macro in the design and is placed by the macro placer. The default type is physical.

-library *lib_name*

Specifies the library in which to create the new design. This is an optional argument. By default, the new design is created in the current library. If specified, the library must be one of the reference libraries for the library which contains the current design.

-target_boundary_area *area*

Specifies the target area of the black box design.

This option is mutually exclusive with **-boundary**.

-boundary *boundary*

Specifies the boundary for the black box design.

This option is mutually exclusive with **-target_boundary_area**.

cell

Specifies a cell whose reference is to be converted to a black box. The list can contain names, patterns, or collections. The cell must be a logical hierarchy cell or a cell with unresolved reference. All other cells in the design with the same reference, would have their reference changed to the black box as well.

DESCRIPTION

When the specified cell has unresolved reference, this command creates a new empty design with the same port interface as the unresolved reference of the cell. The reference of the cell and all other cells with the same unresolved reference is changed to the new design. Any other cells in designs linked to this design as child designs, or designs where this design is linked with the same unresolved reference, are updated as well.

When the specified cell is a logical hierarchy cell, the **create_blackbox** command converts the cell into a physical block and creates a new design with the content of the original cells logical hierarchy. The reference of the cell and all other cells with the same reference is changed to the new design.

The new design created by this command is a black box typed design.

To query the instances that are unbound, use the is_unbound attribute. For empty hierarchies, use the is_empty attribute.

After creating the black box, use the design_type==black_box filter expression to search for physical black boxes. Use the is_logical_black_box attribute for macro style black boxes.

EXAMPLES

The following example creates a new design in the test library by using the port interface and name of the reference of the cell p0. The reference of this cell is unresolved at this point.

```
prompt> create_blackbox -library test p0
1
```

The following example commits the logical hierarchy cell p0 to a physical block and marks the reference block design as black box. The black box design's target_boundary_area is set to 10000.

```
prompt> create_blackbox -target_boundary_area 10000 p0
1
```

SEE ALSO

get_cells(2)

create_blackbox_clock_network_delay

Specifies the clock network delay for clock input ports or blackbox clocks.

SYNTAX

status **create_blackbox_clock_network_delay**
[-max | -min]
[-modes *mode_list*]
[-corners *corner_list*]
-value *delay_value*
from_objects

Data Types

mode_list list
corner_list list
delay_value float
from_objects ports or blackbox clocks

ARGUMENTS

-max

Specifies that the network delay is for maximum delay (setup analysis) only. This option is mutually exclusive with **-min**. By default, when neither **-max** nor **-min** is specified, the network delay is applied to both min (hold) and max (setup).

-min

Specifies that the network delay is for minimum delay (hold analysis) only. This option is mutually exclusive with **-max**. By default, when neither **-max** nor **-min** is specified, the network delay is applied to both min (hold) and max (setup).

-modes *mode_list*

Specifies the list of modes for which to apply the network delay. By default, the clock network delay is applied to all modes.

-corners *corner_list*

Specifies the list of corners for which to apply the network delay. By default, the clock network delay is applied to all corners.

-value *delay_value*

Specifies the network delay value.

from_objects

Specifies the input clock ports or blackbox clocks for which to apply the network delay. The specified ports must already be

identified as clock ports with the **set_blackbox_clock_port** command. The specified blackbox clocks must already be create by the **create_blackbox_generated_clock** command.

DESCRIPTION

This command specifies clock network delay inside of the current block from the specified clock input ports or blackbox clocks. Clock ports are specified with the **set_blackbox_clock_port** command and blackbox clocks are create by the **create_blackbox_generated_clock** command before running the **create_blackbox_clock_network_delay** command.

EXAMPLES

The following example specifies that the maximum clock network delay from port *clk* is 1.2 for max and the minimum clock network delay is 1.0.

```
prompt> create_blackbox_clock_network_delay -max -value 1.2 clk
prompt> create_blackbox_clock_network_delay -min -value 1.0 clk
```

SEE ALSO

commit_blackbox_timing(2)
set_blackbox_clock_port(2)
create_blackbox_generated_clock(2)
get_blackbox_generated_clocks(2)

create_blackbox_constraint

Creates setup and hold constraints for specified ports or blackbox clocks.

SYNTAX

status **create_blackbox_constraint**
-from *from_objects*
-edge rise | false
-to *ports* | -rise_to *ports* | -fall_to *ports*
[-setup | -hold]
[-modes *mode_list*]
[-corners *corner_list*]
[-clock *clock*]
-value *delay_value*

Data Types

from_objects port objects or blackbox clocks
mode_list list of modes
corner_list list of corners
clock clock object
delay_value float

ARGUMENTS

-from *from_objects*

Specifies the clock ports or blackbox clocks which the constraint is from.

-edge rise | fall

Specifies whether the constraint is from the rising edge or falling edge of the clock signal at the specified clock ports.

-to *ports*

Specifies the signal ports that the constraint is for. This creates a constraint for both rise and fall signal at the ports.

-rise_to *ports*

Specifies the signal ports that the constraint is for. This creates a constraint for the rising signal at the ports.

-fall_to *ports*

Specifies the signal ports that the constraint is for. This creates a constraint for the falling edge signal at the ports.

-setup

Specifies that this is a setup constraint. This option is mutually exclusive with the **-hold** option. By default, when neither **-setup** nor **-hold** is specified, the constraint applies to both setup and hold.

-hold

Specifies that this is a hold constraint. This option is mutually exclusive with the **-setup** option. By default, when neither **-setup** nor **-hold** is specified, the constraint applies to both setup and hold.

-modes *mode_list*

Specifies the list of modes what the constraint is for. By default, the constraint is applied to all modes.

-corners *corner_list*

Specifies the list of corners that the constraint is for. By default, the constraint is applied to all corners.

-clock *clock*

Specifies that the constraint time value is the percentage of the given clock's period. The final value of the constraint is the clock period multiplied by the value specified with **-value** option. By default, the value specified with **-value** option is the absolute value for the constraint.

-value *delay_value*

Specifies the delay value for the constraint. When **-clock** is specified, this value means the constraint's value is a certain percentage of the clock period. Without **-clock**, the specified value is the final value for the constraint.

DESCRIPTION

This command defines setup and hold constraints from the specified edge of clock signal at the specified clock ports or blackbox clocks to the specified signal ports. The "from_objects" are clock ports defined by the **set_blackbox_clock_port** command or blackbox clocks defined by the **create_blackbox_generated_clock** command. The "to" ports can be either input or inout signal ports.

The value of the constraint can be described as an absolute value or as a percentage of a specified clock's period.

Multicorner-Multimode Support

EXAMPLES

The following example defines constraints for signal rise at port *$data* from the clock's rising edge at clock port *clk*. The setup constraint value is 0.6. And the hold constraint value is 30 percent of main_clock's period.

```
prompt> create_clock -name main_clock -period 10 [get_ports clk]
prompt> set_blackbox_clock_port clk
prompt> create_blackbox_constraint -from clk -edge rise -to $data -setup -value 0.6
prompt> create_blackbox_constraint -from clk -edge rise -to $data \
  -clock main_clock -value 0.3 -hold
```

SEE ALSO

commit_blackbox_timing(2)
create_blackbox_delay(2)
set_blackbox_clock_port(2)
create_blackbox_generated_clock(2)
get_blackbox_generated_clocks(2)

create_blackbox_delay

Specifies delay paths between the specified ports or blackbox clocks.

SYNTAX

status **create_blackbox_delay**
-from *from_objects* | -rise_from *from_objects* | -fall_from *from_objects*
-to *ports* | -rise_to *ports* | -fall_to *ports*
[-max | -min]
[-modes *mode_list*]
[-corners *corner_list*]
[-clock *clock*]
-value *delay_value*

Data Types

from_objects port objects or blackbox clocks
mode_list list of modes
corner_list list of corners
clock clock object
delay_value float

ARGUMENTS

-from *from_objects*

Specifies the originating ports or blackbox clocks for the timing paths. This option is mutually exclusive with the **-rise_from** and **-fall_from** options. This option creates timing paths for both the rising and falling edges from the specified ports or blackbox clocks.

-rise_from *from_objects*

Specifies the originating ports or blackbox clocks for the rising edge for the timing paths. This option is mutually exclusive with the **-from** and **-fall_from** options. This option creates timing paths for the rising edge to the specified ports or blackbox clocks.

-fall_from *ports*

Specifies the originating ports or blackbox clocks for the falling edge for the timing paths. This option is mutually exclusive with the **-from** and **-rise_from** options. This option creates falling timing paths for the specified ports or blackbox clocks.

-to *ports*

Specifies the destination ports for the timing paths. This option is mutually exclusive with the **-rise_to** and **-fall_to** options. This option creates timing paths for both the rising and falling edges to the specified ports.

-rise_to *ports*

Specifies the destination ports for the rising edge for the timing paths. This option is mutually exclusive with the **-to** and **-fall_to** options. This option creates timing paths for the rising edge to the specified ports.

-fall_to ***ports***

Specifies the destination ports for the falling edge for the timing paths. This option is mutually exclusive with the **-to** and **-rise_to** options. This option creates timing paths for the falling edge to the specified ports.

-max

Specifies that the value is for the maximum delay (setup analysis) only. This option is mutually exclusive with the **-min** option. By default, when neither **-max** nor **-min** is specified, the value applies to both minimum (hold) and maximum (setup).

-min

Specifies that the value is for the minimum delay (hold analysis) only. This option is mutually exclusive with the **-max** option. By default, when neither **-max** nor **-min** is specified, the value is for both minimum (hold) and maximum (setup).

-modes ***mode_list***

Specifies the list of modes that the value applies to. By default, the delay path value is applied to all modes.

-corners ***corner_list***

Specifies the list of corners that the value applies to. By default, the delay path value is applied to all corners.

-clock ***clock***

Specifies that the value is the percentage of the given clock's period. The final delay value for the path is the clock period multiplied by the value specified with **-value** option. By default, the value specified with **-value** option is the absolute delay value for the paths.

-value ***delay_value***

Specifies the path delay value. When **-clock** is specified, this value means that the path delay is the specified percentage of the clock period. If the **-clock** option is not specified, the *delay_value* is the delay value for the paths.

DESCRIPTION

This command defines a set of paths between a set of ports or blackbox clocks and a set of ports as part of the black box timing information. The "from_objects" can be ports or blackbox clocks. The ports can be either input ports or inout ports and can be signal ports or clock ports. The blackbox clocks are created by the **create_blackbox_generated_clock** command. The "to" ports can be either output inout signal ports. Clock ports are specified with the **set_blackbox_clock_port** command before running this command.

The delay value for a path can be described as an absolute value or as a percentage of a specified clock's period.

For a partial netlist, the "to" ports must be ports with no block internal net connection in the original design netlist.

Multicorner-Multimode Support

EXAMPLES

The following example defines a path that rises from port *DataIn* and rises to port *DataOut* with a delay of 0.6 for setup analysis and 0.5 for hold analysis.

```
prompt> create_blackbox_delay -rise_from DataIn -rise_to DataOut -value 0.6 -max
prompt> create_blackbox_delay -rise_from DataIn -rise_to DataOut -value 0.5 -min
```

The following example defines a path from the rising edge of main_clock to the rising or falling edge of port *DataOut*. The delay value is 30 percent of the clock period for main_clock.

```
prompt> create_clock -name main_clock -period 10 [get_ports ClkIn]
prompt> set_blackbox_clock_port ClkIn
prompt> create_blackbox_delay -rise_from ClkIn -to DataOut -clock main_clock -value 0.3
```

SEE ALSO

create_blackbox_constraint(2)
commit_blackbox_timing(2)
set_blackbox_clock_port(2)
create_blackbox_generated_clock(2)
get_blackbox_generated_clocks(2)

create_blackbox_drive_type

Creates an output drive type for black box timing.

SYNTAX

status **create_blackbox_drive_type**
-lib_cell *lib_cell_object*
[-input_transition_rise *transition_time*]
[-input_transition_fall *transition_time*]
name

Data Types

lib_cell_object string
transition_time float
name string

ARGUMENTS

-lib_cell *lib_cell_object*

Specifies the buffer or inverter library cell as the driver cell for this drive type.

-input_transition_rise *transition_time*

Specifies the rise transition time at the input pin of the driver cell. The default is zero.

-input_transition_fall *transition_time*

Specifies the fall transition time at the input pin of the driver cell. The default is zero.

name

Specifies the name for this drive type. Commands such as **set_blackbox_port_drive** can use this name to identify the drive type.

DESCRIPTION

This command creates an output port drive type as part of the black box timing information for the current block.

The drive type created by this command can be used by the **set_blackbox_port_drive** command to describe the driver of a particular port.

Multicorner-Multimode Support

EXAMPLES

The following example defines the driver at output port *DataOut* as a cell whose reference library cell is *BUF8* in reference library *stdreflib1* with 0.01ns rise and fall transition time at the input pin.

```
prompt> create_blackbox_drive_type -lib_cell [get_lib_cells stdreflib1/BUF8] \
      -input_transition_rise 0.01 -input_transition_fall 0.01 driveType1
prompt> set_blackbox_port_drive -type driveType1 [get_ports DataOut]
```

SEE ALSO

create_blackbox_load_type(2)
commit_blackbox_timing(2)
set_blackbox_port_drive(2)
set_blackbox_port_load(2)

create_blackbox_generated_clock

Creates a blackbox generated clock object.

SYNTAX

collection **create_blackbox_generated_clock**
-name *clock_name*
[-source *master_pin*]
[-divide_by *divide_factor* | -multiply_by *multiply_factor* |
[-invert]
-master_clock *clock*

Data Types

clock_name string
master_pin list
divide_factor int
multiply_factor int
clock string

ARGUMENTS

-name *clock_name*

Specifies the name of the blackbox generated clock. The name must have not been used by other clocks. This option is required.

-source *master_pin*

Specifies the master clock source (a clock source pin in the design) from which to derive the clock waveform.

-divide_by *divide_factor*

Specifies the frequency division factor. If the *divide_factor* value is *2*, the blackbox generated clock period is twice as long as the master clock period. This option cannot be used together with option **-multiply_by**.

-multiply_by *multiply_factor*

Specifies the frequency multiplication factor. For example, if the *multiply_factor* value is *3*, the blackbox generated clock period is one-third as long as the master clock period. This option cannot be used together with option **-divide_by**.

-invert

Inverts the blackbox generated clock signal in the case of frequency multiplication and division.

-master_clock *clock*

Specifies the master clock to be used for this blackbox generated clock. This option is required.

DESCRIPTION

Creates a blackbox generated clock object in the blackbox timing model. The blackbox generated clock object is available for use with the **create_blackbox_delay**, **create_blackbox_constraint**, and **create_blackbox_clock_network_delay** commands. Use the **get_blackbox_generated_clocks** command to get the blackbox generated clock objects.

The command can specify the clock source from which it is generated. You must specify the master clock for this generated clock. If the source pin/port is not specified, the specified master clock must have a source pin/port.

The blackbox generated clock can be created as a frequency divided clock (by using the **-divide_by** option), frequency multiplied clock (by using the **-multiply_by** option), or special divide by one (by setting *divide_factor* or *multiply_factor* as 1). The frequency divided or frequency multiplied clock can be inverted by using the **-invert** option.

EXAMPLES

The following example creates a frequency divide_by 2 blackbox generated clock.

```
prompt> create_clock -period 1 -name CLK CLK
prompt> create_blackbox_generated_clock -name GCK -divide_by 2 \
  -source CLK -master_clock CLK
prompt> get_blackbox_generated_clocks GCK
{GCK}
```

SEE ALSO

create_blackbox_delay(2)
create_blackbox_constraint(2)
create_blackbox_clock_network_delay(2)
get_blackbox_generated_clocks(2)

create_blackbox_load_type

Creates an input load type for black box timing.

SYNTAX

status **create_blackbox_load_type**
-lib_cell *lib_cell_object*
name

Data Types

lib_cell_object string
name string

ARGUMENTS

-lib_cell *lib_cell_object*

Specifies the buffer or inverter library cell as the load cell for this load type.

name

Specifies the name for this load type. Commands such as **set_blackbox_port_load** can use the name to identify the load type.

DESCRIPTION

This command creates an input port load type as part of the black box timing information for the current block.

The load type created by this command can be used by the **set_blackbox_port_load** command to describe the load of a particular port.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example defines the load at input port *DataIn* as being five *BUF8* cells from the reference library *stdreflib1*.

```
prompt> create_blackbox_load_type -lib_cell [get_lib_cells stdreflib1/BUF8] loadType1
prompt> set_blackbox_port_load -type loadType1 -factor 5 [get_ports DataIn]
```

SEE ALSO

create_blackbox_drive_type(2)
commit_blackbox_timing(2)
set_blackbox_port_drive(2)
set_blackbox_port_load(2)

create_block

Creates a new block or overwrite an existing block in a module library.

SYNTAX

collection **create_block**
[-force]
block_name

Data Types

block_name string

ARGUMENTS

-force

This option is used to overwrite the block if it already exists in the target library.

block_name

The name of the block to create, with optional library, label, and view specifications. This is specified as follows:

[libName:]blockName[/labelName][.viewName]

If the library specification is not present, the **current_lib** is used. If the label specification is not present, the default un-named label is used. If the view specification is not present, then design is used.

DESCRIPTION

This command creates a new block with an empty top module. It can also be used to re-initialize an existing block into an empty one. The top module name will match the new or re-initialized block. The top module of the block is made the current block. The open-mode will be set to "new". The open_count is (re-)set to 1.

This command returns a collection of the newly created block if successful, or "" if not. If an illegal name ('/', or empty space) is given, then a TCL error is raised.

EXAMPLES

The following example creates a block "top" in library "r4000":

```
prompt> create_block r4000:top
{"r4000:top.design"}
```

SEE ALSO

close_blocks(2)
copy_block(2)
current_block(2)
current_design(2)
get_blocks(2)
get_designs(2)
move_block(2)
open_block(2)
open_lib(2)
remove_blocks(2)
reopen_block(2)
save_block(2)

create_bond_pad_array

Creates a bond pad (hybrid bond link) array in the current design.

SYNTAX

int **create_bond_pad_array**
-layer *layer_name*
[-name *array_name*]
-width *width*
-height *height*
[-bbox { {*llx lly*} {*urx ury*} }]
[-boundary *polyrect*]
-delta {*dx dy*}
[-pattern inline | staggered_1 | staggered_2]
[-repeat {*columns rows*}]
[-origin {*x y*}]

Data Types

layer_name string
array_name string
width float
height float
llx float
lly float
urx float
ury float
polyrect list
dx float
dy float
columns integer
rows integer
x float
y float

ARGUMENTS

-layer *layer_name*

Specifies the bond pad array layer. This is a required option.

-name *array_name*

Specifies the bond pad array name.

-width *width*

Specifies the bond pad shape width. This is a required option.

-height *height*

Specifies the bond pad shape height. This is a required option.

-bbox { {*llx lly*} {*urx ury*} }

Specifies the region in which to place the bond pad. This option and the **-boundary** option are mutually exclusive. If neither this option nor **-boundary** option is used, the entire design area is used.

-boundary *polyrect*

Specifies a *polyrect* region in which to place the bond pad. This option and the **-bbox** option are mutually exclusive. If neither this option nor **-bbox** option is used, the entire design area is used. The *polyrect* is described as a list of location coordinates.

-delta {*dx dy*}

Specifies the horizontal pitch (*dx*) between adjacent columns and the vertical pitch (*dy*) between adjacent rows. This is a required option.

-pattern inline | staggered_1 | staggered_2

Specifies the placement pattern of the bond pad. Valid values for this option are: inline, staggered_1, and staggered_2. The inline argument places bond pad at every grid point. The staggered_1 argument places bond pad only at grid points where the sum of column index and row index is even. The staggered_2 argument places bond pad only at grid points where the sum of column index and row index is odd. If this option is not specified, the command creates an inline bond pad array.

-repeat {*columns rows*}

Specifies the total number of columns and rows in the bond pad array. The command does not create bond pad outside of the bond pad array bounding box, even if you specify a column or row count beyond the bounding box. If this option is not specified, the command fills the bounding box or design area with bond pads.

-origin {*x y*}

Specifies the spacing between the lower-left corner of bond pad array bounding box and the lower-left corner of the bounding box of the bond pad located at column 0 and row 0. If this option is not specified, the command derives the origin so that the bond pad array is centered in the bounding box.

DESCRIPTION

This command creates an array of bond pads.

EXAMPLES

The following example creates an array of bond pads.

```
prompt> create_bond_pad_array -layer AP -width 20 -height 20 \
  -delta {200 200} -bbox {{350 400} {1350 1400}}
```

SEE ALSO

assign_3d_interchip_nets(2)
assign_tsv(2)
check_3d_design(2)
create_3d_mirror_bumps(2)
create_bump_array(2)
create_tsv_array(2)
disconnect_3d_bumps(2)
propagate_3d_connections(2)
propagate_3d_matching_types(2)
set_cell_location(2)

create_bound

Creates a move bound or group bound in the current design.

SYNTAX

collection **create_bound**
-name *bound_name*
[-boundary *regions*
[-dimensions *bound_dimension*]
[-diamond *central_object*]
[-effort low | medium | high | ultra]
[-type soft | hard]
[-exclusive]
[-macro_group]
[-hierarchical_only]
[-design_type *design_type*]
[-repelling diamond | rect]
[-cell *cell*]
[*cell_and_port_and_group_list*]

Data Types

bound_name string
regions list
bound_dimension list
central_object collection
design_type string
cell collection
cell_and_port_and_group_list list

ARGUMENTS

-name *bound_name*

Specifies the name of the bound.

-boundary *regions*

Creates move bounds with the specified boundaries. You can specify one or more rectangles, polygons, and geometric objects. If you specify more than one placeable area, the coarse placer can place cells in any of the move bound's bound shapes. These placeable areas can overlap or be disjoint.

To specify a rectangle, use the following format to specify the lower-left and upper-right corners of the rectangle:

{{*llx lly*} {*urx ury*}}

To specify a polygon, use the following format to specify the coordinates of the polygon:

{{*x1 y1*} {*x2 y2*} {*x3 y3*} ... {*xn yn*}}

Each rectangle and polygon defines a target placement area for the objects. The areas can overlap or be disjoint. The coordinates are relative to the chip origin.

You can also specify polygons as the combined area of a heterogeneous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. For poly_rects, geo_masks, shapes, or other physical objects, the resulting area includes the areas of each object. For layers, the resulting area includes the area of every shape in the layer.

This option is mutually exclusive with the **-dimensions** option; you can specify only one.

-dimensions *bound_dimension*

Creates a group bound with the specified dimensions.

For a group bound, specify the dimension as {*width height*}.

For a diamond bound, specify the dimension as {*length*}. When you use this format, you must also specify the **-diamond** option.

This option is mutually exclusive with the **-boundary** option; you can specify only one.

-diamond *central_object*

Creates a diamond bound centered on the specified object, which can be a port, cell, or pin. You can specify only one object.

The objects in the bound are constrained to lie within the distance specified by the **-dimensions** option (measured as a Manhattan distance) of the specified object. A diamond bound is always a soft bound.

If you specify this option, you must also specify the **-dimensions** option.

-effort low | **medium** | **high** | **ultra**

Specifies the effort used by the tool to keep the cells within in a soft bound.

If you set the effort to **ultra**, the cells are kept in the bound very strictly, with almost no part of any cell . Lower effort levels allow cells to be pushed or pulled beyond the edge of the bound, effort is useful when you want most cells to be contained, but you want to allow placement flexibility to fix timing, wire length, or congestion. Most large bounds work well with **medium** effort.

If you specify the **-boundary** or **-dimensions** option, the default effort is **ultra**. Otherwise, the default effort is **medium**.

You cannot specify the **-effort** option with the **-diamond**, **-repelling**, **-exclusive**, or **-type hard** options.

-type soft | **hard**

Sets the bound type to be either hard or soft.

The default is **soft**.

When you specify **-type hard**,

- You must also specify the **-boundary** or **-dimensions** option
- You cannot specify the **-effort** option

-exclusive

Sets the bound to be exclusive. This option is allowed only for move bounds. A move bound that is exclusive requires all of its cells to be placed inside it and prohibits the placement of other cells in the same area. Exclusive move bounds are respected by both coarse placement and legalization.

-macro_group

Sets the bound to be macro_group bound. This option is allowed only for move bounds.

This option specifies that the bound can contain only cells whose reference is of **macro** design type.

This option is mutually exclusive with **-exclusive**, **-hierarchical_only** and **-design_type** options.

-hierarchical_only

Specifies that the bound can contain only cells that represent module boundary objects, which are created by the **explore_logic_hierarchy** command. This option is allowed only for move bounds. No other cells or ports can be added to move bounds of this type.

This option creates move bounds that are used during floorplanning; the have no effect on other placement or legalization.

-design_type *design_type*

Specifies that the bound can contain only cells whose reference is of the specified design type. This option is allowed only for move bounds. The valid values for this option are **abstract**, **analog**, **black_box**, **corner**, **cover**, **diode**, **end_cap**, **feedthrough**, **fill**, **filler**, **flip_chip_driver**, **flip_chip_pad**, **lib_cell**, **macro**, **module**, **pad**, **pad_spacer**, and **well_tap**. No other cells or ports can be added to move bounds of this type.

This option creates move bounds that are used during floorplanning; they have no effect on other placement or legalization. The expected usage is to create move bounds that contain only hard macros.

-repelling diamond | rect

Specifies that the bound is repelling rather than attractive. Repelling bounds keep the bound contents spread apart by the specified shape region.

-cell *cell*

Specifies the physical cell in which to add the bound. The bound is created in the cell's reference block using the coordinate system of the cell's top block. The cell must reference a block, not a library cell, unless this command is run in the library manager. In the library manager, bounds can be created in library cells.

If you do not specify this option, the bound is created in the current block.

cell_and_port_and_group_list

Specifies the cells, ports, and groups to assign to the bound. The list can contain names, patterns, or collections. Group objects are only valid when creating a repelling bound. Furthermore, repelling bounds may contain either cells and ports or groups, but not a mix.

It is an error to specify a port that has already been assigned to another bound.

If this bound is a move bound, it is an error to specify a cell that has already been explicitly assigned to another move bound.

If this bound is a group bound, it is an error to specify a cell that has already been explicitly assigned to another group bound.

If a cell is hierarchical, its child cells are also implicitly in the bound, as well as any cells added to its hierarchy in the future, unless the child cell is assigned to another bound.

Cells that belong to a relative placement group cannot be individually assigned to a bound. All cells of a relative placement group must be assigned to a bound together.

If you do not specify this argument, the tool creates an empty bound. You can later assign cells and ports to the bound by using the **add_to_bound** command.

DESCRIPTION

This command defines region-based placement constraints for coarse placement. There are two different types of bounds:

- Move bound

  A move bound restricts the placement of cells within fixed regions (bound shapes) of the core area. To create a move bound, specify the absolute coordinates with the **-boundary** option.

- Group bound

  A group bound restricts the placement of cells within a floating region. Cells are placed within a bound but its absolute coordinates are not fixed, instead they are optimized by the placer.

  To specify the floating region size for a group bound, use the **-dimensions** option. If you do not specify the **-dimensions** (or **-boundary**) option, the tool creates an automatic group bound, whose floating region is automatically determined by the placer.

Repelling bounds function like group bounds, but instead of defining an area within which the objects should be placed, they indicate an area of separation that should exist between each pair of objects in the bound.

Generally, there is no guarantee that cells will be placed completely within a bound. For example, coarse placement will violate bounds if the quality of its primary placement objectives would otherwise be degraded. In these situations, you should revisit your and floorplan to ensure that the block is not overconstrained. Alternatively, you can use the **-type hard** option to create a hard bound (the default is soft). The coarse placer tries to honor hard bounds as hard constraints while sacrificing other objectives, such as timing and routability. You should not use many hard bounds because this might result in inferior placement solutions.

To assign cells and ports to an existing bound, use the **add_to_bound** command. To unassign cells and ports from a bound, use the **remove_from_bound** command.

EXAMPLES

The following example creates a hard exclusive move bound named "movebound1" with one rectangular and one polygon bound shape. The bound contains the invA1 cell and reset port.

```
prompt> create_bound -name "movebound1" -boundary {{{100 100} {200 200}} \
  {{1000 1000} {2000 1000} {2000 4000} {1500 4000} {1500 2000} {1000 2000}}} \
  -type hard -exclusive {invA1 reset}
```

The following example creates a move bound named "movebound2" that contains the hierarchical cell named mid. It also implicitly contains all cells within the mid hierarchical cell.

```
prompt> create_bound -name "movebound2" -boundary {{0 0} {200 200}} \
  [get_cells mid]
```

The following example creates a move bound named "movebound3" with rectilinear bound shapes that have the same boundaries as the merged areas of voltage_area_shapes whose names start with VA_SHP_.

```
prompt> create_bound -name "movebound3" \
  -boundary [get_voltage_area_shapes VA_SHP_*]
```

The following example creates an automatic group bound named "groupbound1" with ultra effort that contains all physical cells whose names start with adder.

```
prompt> create_bound -name "groupbound1" -effort ultra { adder* }
```

The following example creates a diamond bound named "temp3"centered on the MOD/U1/A pin. The MOD/INST1 and MOD/INST2 cells are constrained to be at most 64 microns of Manhattan distance away from MOD/U1/A.

```
prompt> create_bound -name "temp3" -dimensions 64 \
  -diamond MOD/U1/A {MOD/INST1 MOD/INST2}
```

The following example creates a macro_group move bound named "macroGroupMovebound".

```
prompt> create_bound -name "macroGroupMovebound" -macro_group -boundary {{0 0} {200 200}}
```

SEE ALSO

add_to_bound(2)
create_bound_shape(2)
get_bound_shapes(2)
get_bounds(2)
remove_bound_shapes(2)
remove_bounds(2)
remove_from_bound(2)
report_bounds(2)

create_bound_shape

Creates a bound shape on a move bound.

SYNTAX

collection **create_bound_shape**
-bound *bound_object*
-boundary { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } |
geometric_objects

Data Types

bound_object	collection
llx	float
lly	float
urx	float
ury	float
x	float
y	float
geometric_objects	collection

ARGUMENTS

-bound *bound_object*

The bound on which to create the bound shape.

-boundary { {*llx lly*} {*urx ury*} } | {{*x y*} {*x y*} {*x y*} {*x y*}...} | geometric_objects

Specifies the boundary of the bound shape. The boundary may be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates (i.e., {*llx lly*} {*urx ury*}). A polygon is specified by its points (i.e., {*x y*} {*x y*} {*x y*} {*x y*}...).

A polygon may also be specified as the combined area of a heterogenous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

When specifying the boundary polygon as a collection of physical objects, the resulting area must resolve to a single, connected polygon. It is an error to specify a collection of objects with a combined area that resolves to multiple polygons.

DESCRIPTION

The create_bound_shape command creates a bounding region on a move bound. A move bound may have multiple bound shapes.

This command will not allow bounding region creation on a macro_group move bound. User can create bound shapes on macro_group move bound while creating the bound itself using create_bound.

EXAMPLES

prompt> **create_bound_shape -bound "movebound1" -boundary {{100 100} {200 200}}**

prompt> create_bound_shape -bound "movebound2" -boundary [get_shapes RECT_0]

SEE ALSO

remove_bound_shapes(2)
get_bound_shapes(2)
create_bound(2)
get_bounds(2)
add_to_bound(2)
remove_from_bound(2)
report_bounds(2)
remove_bounds(2)

create_boundary_cells

Creates and places boundary cells into a design.

SYNTAX

status **create_boundary_cells**
[-left_boundary_cell *lib_cell_name*]
[-right_boundary_cell *lib_cell_name*]
[-bottom_boundary_cells *lib_cell_name*]
[-top_boundary_cells *lib_cell_name*]
[-bottom_left_outside_corner_cell *lib_cell_name*]
[-bottom_right_outside_corner_cell *lib_cell_name*]
[-top_left_outside_corner_cell *lib_cell_name*]
[-top_right_outside_corner_cell *lib_cell_name*]
[-bottom_left_inside_corner_cells *lib_cell_name*]
[-bottom_right_inside_corner_cells *lib_cell_name*]
[-top_left_inside_corner_cells *lib_cell_name*]
[-top_right_inside_corner_cells *lib_cell_name*]
[-mirror_left_outside_corner_cell]
[-mirror_right_outside_corner_cell]
[-mirror_left_inside_corner_cell]
[-mirror_right_inside_corner_cell]
[-mirror_left_boundary_cell]
[-mirror_right_boundary_cell]
[-mirror_left_inside_horizontal_abutment_cell]
[-mirror_right_inside_horizontal_abutment_cell]
[-tap_distance *distance*]
[-top_tap_cell *lib_cell_name*]
[-bottom_tap_cell *lib_cell_name*]
[-prefix *boundary_cell_prefix*]
[-separator *boundary_cell_separator*]
[-insert_into_blocks]
[-at_va_boundary]
[-no_1x]
[-min_row_width *width*]
[-add_metal_cut_allowed]
[-do_not_swap_top_and_bottom_inside_corner_cell]

Data Types

boundary_cell_prefix string
boundary_cell_separator string
lib_cell_name string
orientation orientation in DEF syntax
distance float
width float
objects list

ARGUMENTS

-left_boundary_cell *lib_cell_name*

Specifies the library cell to be placed at the beginning of each cell row.

-right_boundary_cell *lib_cell_name*

Specifies the library cell to be placed at the end of each cell row.

-bottom_boundary_cells *lib_cell_name*

Specifies the library cells to be placed along the bottom object boundaries. In a double-back design, adjacent rows are flipped; the bottom boundary cell is used on unflipped bottom boundary rows and on flipped top boundary rows.

-top_boundary_cells *lib_cell_name*

Specifies the library cells to be placed along the top of object boundaries. In a double-back design, adjacent rows are flipped; the top boundary cell is used on unflipped top boundary rows and on flipped bottom boundary rows.

-bottom_left_outside_corner_cell *lib_cell_name*

Specifies the library cell to be placed at bottom-left outside corners. An outside corner is a corner with a 90-degree inside angle. A bottom corner is a corner that is on a bottom boundary. In a double-back design, adjacent rows are flipped; the bottom corner cell is used on unflipped bottom boundary rows and on flipped top boundary rows.

-bottom_right_outside_corner_cell *lib_cell_name*

Specifies the library cell to be placed at bottom-right outside corners. An outside corner is a corner with a 90-degree inside angle. A bottom corner is a corner that is on a bottom boundary. In a double-back design, adjacent rows are flipped; the bottom corner cell is used on unflipped bottom boundary rows and on flipped top boundary rows.

-top_left_outside_corner_cell *lib_cell_name*

Specifies the library cell to be placed at top-left outside corners. An outside corner is a corner with a 90-degree inside angle. A top corner is a corner that is on a top boundary. In a double-back design, adjacent rows are flipped; the top corner cell is used on unflipped top boundary rows and on flipped bottom boundary rows.

-top_right_outside_corner_cell *lib_cell_name*

Specifies the library cell to be placed at top-right outside corners. An outside corner is a corner with a 90-degree inside angle. A top corner is a corner that is on a top boundary. In a double-back design, adjacent rows are flipped; the top corner cell is used on unflipped top boundary rows and on flipped bottom boundary rows.

-bottom_left_inside_corner_cells *lib_cell_name*

Specifies the library cell to be placed at the bottom-left of inside corners. An inside corner is a corner with a 270-degree inside angle. A left inside corner cell is an inside corner cell on the left end of the horizontal edge that makes up the inside corner.

-bottom_right_inside_corner_cells *lib_cell_name*

Specifies the library cell to be placed at the bottom-right of inside corners. An inside corner is a corner with a 270-degree inside angle. A right inside corner cell is an inside corner cell on the right end of the horizontal edge that makes up the inside corner.

-top_left_inside_corner_cells *lib_cell_name*

Specifies the library cell to be placed at the top-left of inside corners. An inside corner is a corner with a 270-degree inside angle. A left inside corner cell is an inside corner cell on the left end of the horizontal edge that makes up the inside corner.

-top_right_inside_corner_cells *lib_cell_name*

Specifies the library cell to be placed at the top-right of inside corners. An inside corner is a corner with a 270-degree inside angle. A right inside corner cell is an inside corner cell on the right end of the horizontal edge that makes up the inside corner.

-mirror_left_outside_corner_cell

Places left outside corner cells in a mirrored orientation.

-mirror_right_outside_corner_cell

Places right outside corner cells in a mirrored orientation.

-mirror_left_inside_corner_cell

Places left inside corner cells in a mirrored orientation.

-mirror_right_inside_corner_cell

Places right inside corner cells in a mirrored orientation.

-mirror_left_boundary_cell

Places left boundary cells in a mirrored orientation.

-mirror_right_boundary_cell

Places right boundary cells in a mirrored orientation.

-mirror_left_inside_horizontal_abutment_cell

Places left inside horizontal abutment cells in a mirrored orientation.

-mirror_right_inside_horizontal_abutment_cell

Places right inside horizontal abutment cells in a mirrored orientation.

-tap_distance *distance*

Specifies the distance in microns between tap cells.

-top_tap_cell *lib_cell_name*

Specifies the tap cell to be placed at the top boundary. The tool inserts this tap cell at the specified interval on top boundary rows. This tap cell is for the boundary cells and should not be confused with tap cells for standard cells. In a double-back design, adjacent rows are flipped; the top tap cell is used on unflipped top boundary rows and on flipped bottom boundary rows. This option must be used with the **-tap_distance** option.

-bottom_tap_cell *lib_cell_name*

Specifies the tap cell to be placed at the bottom boundary. The tool inserts this tap cell at the specified interval on bottom boundary rows. This tap cell is for the boundary cells and should not be confused with tap cells for standard cells. In a double-back design, adjacent rows are flipped; the bottom tap cell is used on unflipped bottom boundary rows and on flipped top boundary rows. This option must be used with the **-tap_distance** option.

-prefix *boundary_cell_prefix*

Specifies the prefix for the created boundary cells. By default, no prefix is added. When you use this option, the command uses the following naming convention for created boundary cells:

boundarycell!*prefix*!*library_cell_name*!*number*

By default, no prefix is added and the command uses the following naming convention:

boundarycell!*library_cell_name*!*number*

-separator *boundary_cell_separator*

Specifies the separator character that is used when composing the instance name of the boundary cell. The boundary cell instance name consists of a prefix, the library reference cell name and an incrementing number. For example, the instance name "boundarycell!MY_CELL!25" has the boundary cell prefix, the MY_CELL library reference cell name, the incrementing number 25, and the boundary cell separator !. This naming convention allows you to do pattern matching selection of the boundary cells.

By default, the separator character is an exclamation mark (!) and the command uses the following naming convention for created boundary cells:

boundarycell![*prefix*!]*library_cell_name*!*number*

-insert_into_blocks

Performs boundary cell insertion in the subblocks. By default, the tool performs boundary cell insertion only in the top-level block.

-at_va_boundary

Places horizontal boundary cells on both sides of the voltage area boundaries. When you use this option, rows are treated as cut by voltage area boundaries. Therefore, boundary cells are placed on both sides of a voltage area boundary. By default, voltage area boundaries are ignored during boundary cell insertion.

-no_1x

Prevents the tool from placing boundary cells on a row when the row length equals two times the corner cell width plus one unit tile width. Note that if the row length equals two times the corner cell width, the tool inserts boundary cells on that row.

This option can be used only when the **-top_right_outside_corner_cell**, **-bottom_right_outside_corner_cell**, **-top_left_outside_corner_cell** and **-bottom_left_outside_corner_cell** options are all specified.

-min_row_width *width*

Skips rows with a row width that is smaller than the specified width value.

-add_metal_cut_allowed

Adds metal cut allowed underneath the standard cell placeable area. Metal cut allowed and forbidden preferred grid extension routing guides are created. Metal cut allowed routing guides cover the area taken up by all the placable site rows reduced by the vertical and horizontal shrink factor. The vertical shrink factor is expressed as a percentage of the smallest site row height and the default is 50%. It can be set using the chipfinishing.metal_cut_allowed_vertical_shrink_factor app option. The horizontal shrink factor is expressed as a percentage of the smallest site row width. It can be set using the chipfinishing.metal_cut_allowed_horizontal_shrink_factor app option. Finally, forbidden preferred grid extension routing guides are created to cover the remaining area up to the boundary.

-do_not_swap_top_and_bottom_inside_corner_cell

Prevents swapping of the top and bottom inside-corner cells for flipped rows.

By default, if a top inside-corner cell is on a flipped row, the tool uses a bottom inside-corner cell instead. If a bottom inside-corner cell is on a flipped row, the tool uses a top inside-corner cell instead.

DESCRIPTION

This command adds boundary cells around the boundaries of objects, such as voltage areas, macros, blockages, and the core area.

EXAMPLES

The following example creates boundary cells on both the left and the right edges of boundaries.

```
prompt> create_boundary_cells -left_boundary_cell myLib/CellLeft \
        -right_boundary_cell myLib/CellRight
```

SEE ALSO

set_boundary_cell_rules(2)
report_boundary_cell_rules(2)
remove_boundary_cell_rules(2)
compile_boundary_cells(2)
check_boundary_cells(2)
compile_targeted_boundary_cells(2)
check_targeted_boundary_cells(2)
create_stdcell_fillers(2)
create_tap_cells(2)
create_targeted_boundary_cells(2)

create_budget_busplan

Create a busplan to define the timing of budget segments

SYNTAX

int **create_budget_busplan**
[-name *name*]
[-rule *net_estimation_rule*]
-from *pins_or_ports*
-nets *nets*
[-force]

Data Types

name string
net_estimation_rule string
pins_or_ports collection
nets collection

ARGUMENTS

-name *name*

Specify the name of the busplan object. If you do not specify a name, a unique name will be generated automatically.

-rule *net_estimation_rule*

Specify a net estimation rule. The rule must have be declared previously with the **set_net_estimation_rule** command. If you do not define a rule, the "default" rule is used. The rule is used when calculating the delay of segments in the budget. In particular, the "ns_per_mm" parameter of the rule will be multiplied by the physical length of the segment to get a delay value.

-from *pins_or_ports*

Specify the pins or ports where the busplan begins. If you specify ports, they must be input ports. If you specify pins, they must be output pins of budgeted blocks. Budgeted blocks should be declared by using the **-add_blocks** option of the **set_budget_options** command.

You must specify one of **-from** and **-nets**.

-nets *nets*

Specify a set of nets through which the busplan should pass. The net must be in the direct fanout of a legal bus start point, separated by buffers or inverters. See **-from** option description for a definition of legal bus start points.

You must specify one of **-from** and **-nets**.

-force

Create the busplan, even if it crosses another busplan. If you use this option, then some or all of the existing busplans are removed and replaced by the new busplan definition. By default, a busplan is not created in a place where another busplan already crosses.

DESCRIPTION

This command creates a database object called a "busplan", that can be used to define the timing of budget segments. A busplan is a collection of paths through the design that start at top-level ports or block outputs and end at top-level ports or block inputs. The paths can cross multiple levels of hierarchy and can traverse the following circuit elements:

- buffers and inverters
- level shifters
- isolation cells
- flip-flops and latches

Busplans are declared by specifying either **-from** or **-nets** to indicate where the bus starts. If a path containing only the given elements leads to a legal end point, the busplan is created. You can specify many start pins, ports or nets at the same time to form a multibit busplan.

The busplan also has a net_estimation_rule object associated with it, which you set by using the **-rule** option. The net estimation rule object must be previously declared by using the **set_net_estimation_rule** command. The rule will be used when calculating the delay of segments in the budget. In particular, the "ns_per_mm" parameter of the rule will be multiplied by the physical length of the segment to get a delay value.

After creating a busplan, the budgeter can extract budgeted "segments" which lie along the path of the busplan. You can use the **-busplans** option of the **report_budget** command to see these budget segments. Each budget segment will represent a connection between budgeted blocks, or a feedthrough path through a budgeted bloc.

Use the **-busplans** option of the **compute_budget_constraints** to apply the timing of your busplans to your budget. See the example below for details.

This command returns 1 on success, 0 otherwise.

EXAMPLES

Create a net estimation rule with 0.5 ns_per_mm propagation time. Define 4 bit busplan starting at block1 and ending at block3. Report on the segments along the busplan. (Note that the bus fed through block2.) Apply the calculated delays from the busplan to the budget.

```
prompt> set outpins B1/OUT1[*]
prompt> set_net_estimation_rule fast_rule -parameter ns_per_mm -value 0.5
prompt> create_budget_busplan -name bus1 -rule fast_rule -from $outpins
prompt> report_budget -busplans bus1
****************************************
Report : report_budget -busplans
 ...
```

```
****************************************

      Segments for busplan 'bus1' (rule: fast_rule)
                                  Calculated Current
From                 To               Delay     Delay
--------------------------------------------------------------------------------
B1/OUT1[3]             B2/IN1[3]          0.136     --
B1/OUT1[2]             B2/IN1[2]          0.136     --
B1/OUT1[1]             B2/IN1[1]          0.136     --
B1/OUT1[0]             B2/IN1[0]          0.136     --
B2/IN1[3]            B2/OUT1[3]           0.554     --
B2/IN1[2]            B2/OUT1[2]           0.554     --
B2/IN1[1]            B2/OUT1[1]           0.554     --
B2/IN1[0]            B2/OUT1[0]           0.554     --
B2/OUT1[3]             B3/IN1[3]          0.522     --
B2/OUT1[2]             B3/IN1[2]          0.522     --
B2/OUT1[1]             B3/IN1[1]          0.522     --
B2/OUT1[0]             B3/IN1[0]          0.522     --
1
prompt> compute_budget_constraints -busplans bus1
```

SEE ALSO

compute_budget_constraints(2)
set_segment_budget_constraints(2)
remove_busplans(2)
report_budget(2)
set_net_estimation_rule(2)
report_net_estimation_rules(2)

create_buffer_trees

Builds buffer trees from the specified driver pins or nets

SYNTAX

status **create_buffer_trees**
[-from *pin_net_list*]
[-incremental]
[-global_route_based]

Data Types

pin_net_list list

ARGUMENTS

-from *pin_net_list*

Specifies the list or collection of driver pins or nets for which the tool will create buffer trees. When the **-from** option is not specified, the **create_buffer_trees** command works on all drivers.

-incremental

Enables incremental buffer tree building from the specified drivers. In the default non-incremental mode, any existing buffer tree downstream from the driver is first removed, and the buffer tree is completely rebuilt. In incremental mode, the existing buffer tree is kept, and buffering is only done on the nets directly driven by the specified drivers.

-global_route_based

Enables global route based buffer tree building (GRopto). By default, buffer trees are built using virtual route topologies and extraction. With the **-global_route_based** option, first a congestion map update is performed, then the global router is used to derive buffer tree topologies and to calculate net RCs when building the tree.

DESCRIPTION

create_buffer_trees is used to build buffer trees from a specified list of drivers. Those drivers are specified by the **-from** option, and can be given as a list or collection of driver pins or nets. If nets are given, then the driver pins of those nets are used as the buffer tree drivers. **create_buffer_trees** can be run on the full design by omitting the **-from** option, but this is not the recommended use model. If you are doing initial buffer tree building after initial coarse placement of the design, or you want to fully rebuild all buffer trees after an incremental coarse placement, the recommended use model is to call **place_opt -from initial_drc -to initial_drc**.

create_buffer_trees is a preroute optimization command, and typically would be used after initial buffer trees are built in the

place_opt initial_drc step, and sometime before routing is performed on the design with **route_auto**. **create_buffer_trees** does not route the buffer trees, and it does not legalize the placement. If you are working with a routed design, more targeted post-route buffering can be done using the **add_buffer_on_route** command.

Typical use models for **create_buffer_trees** include:

- Custom building specific buffer trees that have unique requirements, like different lib_cells or routing layers.
- Repairing buffers trees that were damaged by manual modifications to the design placement, such as a user Tcl script that tweaks the placement of registers or other categories of standard cells.

By default, **create_buffer_trees** does full buffer tree building downstream of the specified drivers. This includes removal of any existing buffers and inverters, then rebuilding of the full tree. With the **-incremental** option, **create_buffer_trees** will do no removal of existing buffers. It will only buffer the nets directly driven by the specified drivers, and won't trace through any pre-existing downstream buffers during tree building.

By default, **create_buffer_trees** uses the virtual router to derive the routing topologies that guide the buffer insertion, and it also uses the current preroute extraction model to estimate RCs on the nets driven by the inserted buffers (the default preroute extraction model is called RDE, or route-driven estimation). If the **-global_route_based** option is used, then **create_buffer_trees** will instead use the global router for the buffering topologies as well as the RCs on the nets. "GRopto" is a term sometimes used to refer to buffer tree building with global routing information. Those global routes are not kept in the design after buffering, they are discarded. Using the **-global_route_based** option also triggers a global route congestion map update, so that the buffering topologies are fully congestion aware. Note that using this option will increase the runtime of the command. **create_buffer_trees** does not honor the **place_opt.initial_drc.global_route_based** app option which triggers GRopto mode in the **place_opt initial_drc** command -- you muse use the **-global_route_based** option to get equivalent behavior in **create_buffer_trees.** In either virtual route or global route mode, the calculated RCs on the nets are influenced by any min or max layer settings or non-default routing rules applied to the nets driven by the buffer trees.

If you are building multiple buffers trees, runtime is improved by calling **create_buffer_trees** one time with a collection or list of those drivers specified by the **-from** option, rather than calling **create_buffer_trees** multiple times in a roe.

create_buffer_trees honors the **set_lib_cell_purpose** constraints, and will only use buffers and inverters that were made available with the *optimization* purpose. If you change the **set_lib_cell_purpose** constraints later in the flow, it is possible for some buffers on those trees to then be resized to lib_cells from the new **set_lib_cell_purpose** constraints.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

place_opt(2)

create_bump_array

Creates an array of bump cell instances.

SYNTAX

int **create_bump_array**
[-name *bump_array_name*]
[-lib_cell *bump_cell_name*]
[-lib_cell_collection *lib_cell_list*]
[-bbox { {*llx lly*} {*urx ury*} }]
[-boundary *polyrect*]
-delta {*dx dy*}
[-pattern inline | staggered_1 | staggered_2]
[-repeat {*columns rows*}]
[-origin {*x y*}]
[-orientation N | W | S | E | FN | FS | FE | FW]

Data Types

bump_array_name string
bump_cell_name string
lib_cell_list collection of library cells
llx float
lly float
urx float
ury float
polyrect list
dx float
dy float
columns integer
rows integer
x float
y float

ARGUMENTS

-name *bump_array_name*

Specifies the bump array name. This name is used to name the bumps within the array. If this option is not used, the array is named default_bump_array_*index*, where *index* is a positive integer. The bump cell instances in the array are named as *bump_array_name*_colIndex_rowIndex, where colIndex and rowIndex are integers that represent the column and row of the bump instance, respectively.

-lib_cell *bump_cell_name*

Specifies the name of bump library cell. All bump instances in a bump array must have the same library cell. User can specify a simple cell name. The tool will search all libraries for the right lib_cell. User can specify the library name and lib_cell name. The library name must be described first followed by lib_cell name. The two names are separated by *:* or */*. User can also specify the view name. The full name of the lib_cell is described as library_name:cell_name.view_name. User can also use hierarchy delimiters, namely library_name/cell_name/view_name. User must describe library name and cell name if they want to describe view name.

-lib_cell_collection *lib_cell_list*

Specifies the collection of a library cell using *get_lib_cells* command. User can also use the full name of a bump lib_cell. This option and *-lib_cell* option are mutually exclusive. If user specify multiple library cells, only the first one is used. User must either use this option or *-lib_cell* option.

-bbox { {*llx lly*} {*urx ury*} }

Specifies the region in which to place the bump cell instances. No bump cell instances or parts of a bump cell instance can extend beyond the specified region. This option and the **-boundary** option are mutually exclusive. If neither this option nor **-boundary** option is used, the entire design area is used.

-boundary *polyrect*

Specifies a *polyrect* region in which to place the bump cell instances. No bump cell instances or parts of a bump cell instance can extend beyond the specified region. This option and the **-bbox** option are mutually exclusive. If neither this option nor **-bbox** option is used, the entire design area is used. The *polyrect* is described as a list of location coordinates.

-delta {*dx dy*}

Specifies the horizontal spacing (*dx*) between adjacent columns and the vertical spacing (*dy*) between adjacent rows. This is a required option.

-pattern inline | staggered_1 | staggered_2

Specifies the placement pattern of the bump array. Valid values for this option are: inline, staggered_1, and staggered_2. The inline argument places bump instances at every grid point. The staggered_1 argument places bump instances only at grid points where the sum of column index and row index is even. The staggered_2 argument places bump instances only at grid points where the sum of column index and row index is odd. If this option is not specified, the command creates an inline bump array.

-repeat {*columns rows*}

Specifies the total number of columns and rows in the bump array. The command does not create bump instances outside of the bump array bounding box, even if you specify a column or row count beyond the bounding box. If this option is not specified, the command fills the bounding box or design area with bump cells.

-origin {*x y*}

Specifies the spacing between the lower-left corner of bump array bounding box and the lower-left corner of the bounding box of the bump cell instance located at column 0 and row 0. The bump array bounding box is either specified by **-bbox** option or the bounding box of the shape specified by **-boundary** option. If neither **-bbox** nor **-boundary** is used, the bump array bounding box is the entire die area.

If you do not use **-origin** but you specify the **-repeat** option, the command derives the origin so that the bump array is centered in the bounding box. It is important that user chooses the column or row count of the **-repeat** option small enough so that the bump array can be entirely inside the bump array bounding box. If the column or row count is too large, the calculated origin can be outside of the bump array bounding box when the bump array is centered in the bounding box. The bumps outside of the bump array bounding box will not be created in the design.

If this option is not specified and **-repeat** option is not selected either, the command uses (0,0) as the origin.

-orientation N | W | S | E | FN | FS | FE | FW

Specifies the orientation of all bumps in the array. If this option is not specified, the default orientation is N.

DESCRIPTION

This command creates an array of bump cell instances.

EXAMPLES

The following command creates an array of bump cells named BUMP_PAD, with horizontal and vertical spacing of 200. The bump cells are created within a bounding box of {{350 400} {2350 2400}}.

```
prompt> create_bump_array -lib_cell BUMP_PAD -delta {200 200} -bbox {{350 400} {2350 2400}}
```

SEE ALSO

create_bump_pattern(2)
place_io(2)

create_bump_block

Creates a reference cell of a bump.

SYNTAX

collection **create_bump_block**
[-force]
-dimensions *dimensions_list*
[-origin *coordinate*]
-layer *layer_name*
-port_name *port_name*
[-library *lib_name*]
bump_name

Data Types

dimensions_list list
coordinate list
layer_name string
port_name string
lib_name string
bump_name string

ARGUMENTS

-force

Overwrites the bump if one with the same name already exists in the target library.

-dimensions *dimensions_list*

Specifies the width and height of the die block in micron.

-origin *coordinate*

Specifies the coordinate of origin.

-layer *layer_name*

Specifies the layer on which the bump rectangle should be created.

-port_name *port_name*

Specifies the port name of the bump.

-library *lib_name*

Specifies the target library that contains the process technology of the bump to be created. If not specified, the bump cell is created in the current library.

bump_name

Specifies the bump name to be created.

DESCRIPTION

The **create_bump_block** command creates a reference cell of a bump that consists of a rectangle at the specified layer with the given port name. This is for quick exploration flow when no actual bump cell is available in the library. For manufacturing, a bump should be properly designed to meet the manufacturing rules.

EXAMPLES

The following example creates a bump named ubump.design in the library top.exp.

```
prompt> create_bump_block ubump -force -dimensions {30 30} -origin {0 0} -layer M9 -port_name BUMP -library top.exp
Information: Creating block 'ubump.design' in library 'top.exp'. (DES-013)
{top.exp:ubump.design}
```

SEE ALSO

create_3d_top_design(2)
create_die_block(2)
create_die(2)
create_lib(2)

create_bump_cluster

Creates a bump cluster in current design based on given pattern cell design, at given coordinate location, and with specified orientation.

SYNTAX

int **create_bump_cluster**
-pattern_cell *pattern_cell_name_or_collection*
-coordinates {*llx lly*}
[-net *net_name*]
[-cluster_name *cluster_name*]
[-orientation R0 | R90 | R180 | R270 | MX | MXR90 | MY | MYR90]
[-coord_spec bl | c]
[-force]

Data Types

pattern_cell_name_or_collection string or collection
llx float
lly float
net_name string
cluster_name string

ARGUMENTS

-pattern_cell *pattern_cell_name_or_collection*

Specifies the pattern cell to be used for creating bump cluster. If a collection is specified, it must contain exactly one pattern cell. This is a required option.

-coordinates {*llx lly*}

Specifies the coordinate location for lower left corner of pattern cell to be placed. Note, if orientation is specified, this is the lower left corner after orientation has been applied. This is a required option.

-net *net_name*

Specifies the net name to be used for connecting all cells inside bump cluster. If specified net does not exist, a new net is created. Default net name is based on bump cluster name, and is of form: *cluster_name*_net.

-cluster_name *cluster_name*

Specifies the name to be used for edit_group containing all cells of bump cluster created by this command. If *cluster_name* is not provided, command generates a unique cluster name of form: bump_cluster_xxx, where xxx is a unique integer suffix.

-orientation R0 | **R90** | **R180** | **R270** | **MY** | **MXR90** | **MX** | **MYR90**

Specifies the orientation of bump cluster as placed at given {*llx lly*} coordinate location. If this option is not specified, the default orientation is R0.

-coord_spec bl | c

Specify the user specified coordinates is for bottom left (bl) or center (c) of the new bump cluster. If this option is not specified, the default coord_spec is bottum left.

-force

Specifies that bump cluster should be created even if it overlaps with an existing bump cluster. By default, a bump cluster is not created if it detects an overlap with an existing bump cluster.

DESCRIPTION

This command creates a bump cluster in current design based on given pattern cell design, at given coordinate location, and with specified orientation.

EXAMPLES

The following example creates a bump cluster.

prompt> **create_bump_cluster -pattern_cell *:TSV4BUMP81 -coordinates {1000 1000}**

SEE ALSO

get_bump_cluster_name(2)
get_bump_cluster_objects(2)
create_tsv_array(2)
create_3d_mirror_bumps(2)

create_bump_pattern

Creates bumps and matching types for a cluster of pads.

SYNTAX

int **create_bump_pattern**
-file_name *pattern_file*
-bump_prefix *prefix*

Data Types

pattern_file string
prefix string

ARGUMENTS

-file_name *pattern_file*

Specifies the name of a text file which describes the bump patterns to be created.

-bump_prefix *prefix*

Specifies the prefix of bump names. The bump names are created by adding an underscore and integer index at the end of the bump prefix. The default prefix is **default_bump_array**.

DESCRIPTION

This command creates one or multiple bumps for some placed IO pads. It also generates one or multiple matching types for impending bump assignment using command **place_io**.

You must create a text file, called a pattern file, to describe the bumps and matching types to be created by this command. The format of this file is described as follows. Specifically, the following line should be used at the beginning of a pattern file to select the bump reference cell. The string **bump_lib_cell** is a keyword and must be placed at the beginning of the line. The **LIB_CELL_NAME** is the full name of the reference cell, including the library name. If a simple string is used as **LIB_CELL_NAME**, current library is assumed.

bump_lib_cell LIB_CELL_NAME

The following line should also be used at the beginning of a pattern file to specify the pitch value of bump rows or columns. In particular, **bump_pitch** is a keyword and must be placed at the beginning of the line. The **DISTANCE** is in microns.

bump_pitch DISTANCE

If the pattern file contains multiple lines which define a bump reference cell or bump pitch, the values later in the file will override the previous values and affect the bumps created afterwards.

Bumps and matching types are created for individual clusters of pads, one cluster at a time. For each pad cluster, which is called an IO group, a single unit of description is used in the pattern file. This unit starts with the keyword **io_group**. There must be a left curly bracket after the keyword **io_group**. The detailed description follows the left curly bracket. The unit ends with a right curly bracket.

io_group { ..._detailed_description... }

In the detailed description part, you must first describe all pads in the cluster. Each pad name must be placed in a pair of curly brackets by itself. Within the same pair of curly brackets, you can add a string type attribute for the pad. The attribute string is separated from the pad name by one or more white space characters. If no attribute string is given, the attribute of the pad is a single letter **s**. All pad names and their own curly brackets are placed in a linear order. The entire order is placed within a pair of curly brackets as follows.

{
{pad1 _string_of_pad1}
{pad2}
}

All pads must be placed before running the **create_bump_pattern** command. The minimum-size rectangle that covers all pads is called the pad boundary box. If the width of the box is larger than the height, the pads are considered to be placed in rows. If the height of the box is larger than the width, the pads are considered to be placed in columns. If the width is the same as the height, the pads are with no orientation.

Bumps to be created and placed are described in groups. Each group of bumps are aligned along a single line. The orientation of the bump group is defined after the pads are described. Specifically, the line starts with a left curly bracket followed by a keyword **orientation**. The value of orientation follows and can be **row**, **column**, or **perpendicular**. The default value is **perpendicular**, which means that the bump line is perpendicular to the pad placement orientation. Clearly, this value cannot be used if pads have no orientation. The line that defines bump group orientation must end with a right curly bracket. The following line shows an example.

{ orientation row }

After the orientation of the bump groups is specified, the bump groups are defined. Each bump group is a single line of bumps whose orientation is defined using aforementioned orientation line. To describe a line of bumps, five elements are used, each of which is placed in a pair of curly brackets. The first element is the distance between the center line of bumps and the center of pad boundary box. Negative values are used when the pad boundary box center is above or at the right side of the bump center line. Positive values are used when the pad boundary box center is below or at the left side of the bump center line. A single bump line is divided into two parts by the pad boundary box. The second and third elements are used to describe the part either on the left or below pad boundary box when bumps are placed in a row or column, respectively. The second element is the distance between the center of the rightmost or top bump and the left or bottom boundary edge of the pad boundary box, respectively. If the value is positive, the bump center is on the left side or below the corresponding edge. The value is negative otherwise. The third element is a sequence of strings, each of which represents a unique bump. The first string represents the rightmost or top bump. The order is from right to left or from top to bottom. The fourth and fifth elements are used to describe the part of bumps either on the right side or above pad boundary box when bumps are placed in a row or column, respectively. The fourth element is the distance between the center of the leftmost or bottom bump and the right or top boundary edge of the pad boundary box, respectively. If the value is positive, the bump center is on the right side or above the corresponding edge. The value is negative otherwise. The fifth element is a sequence of strings, each of which represents a unique bump. The first string represents the leftmost or bottom bump. The order is from left to right or from bottom to top.

The two parts of a line of bumps have the same pitch, which is defined by **bump_pitch** line outside of current **io_group** unit. However, you can add a sixth element to define a different pitch value for and only for the current bump line.

Multiple lines can be used to describe multiple rows or columns of bumps within an **io_group** unit. The full path name of the first pad in the pad description is used as the name of the IO group. The bump names are created by adding an underscore and integer index at the end of the bump prefix.

Bumps to be created have string attributes. Pads also have string attributes. The bumps and pads with the same attribute string form a single matching type, which is created by the **create_bump_pattern** command. The name of the matching type is created by

combining the IO group name, an underscore character, and the string of attribute. However, the hierarchy delimiter / is replaced by an underscore _. The uniquify number is created based on the ratio of bumps and pads in the matching type.

A pattern file can contain comment lines. A comment line starts with a **#** symbol. In addition, a two-character string // indicates all characters afterwards are comments.

EXAMPLES

A sample pattern file is shown below. The second and third elements are empty pairs of curly brackets, indicating that all four bumps are on the right side of the pad boundary box. There is no bump on the left side. **pad1** and **pad2** have an attribute equal to **s**. Three matching types will be created, for **vdd**, **gnd**, and **s**, respectively.

```
bump_lib_cell BUMP
bump_pitch  200
io_group {
{ {pad1} {pad2} {pad3 vdd} {pad4 gnd} }
{ orientation row }
{ {0} {} {} {60} {vdd gnd s s} }
}
```

SEE ALSO

create_bump_array(2)
create_matching_type(2)

create_bundle

Creates a new bundle of nets in the design.

SYNTAX

collection **create_bundle**
[-name *bundle_name*]
[*objects*]

Data Types

bundle_name string
objects objects

ARGUMENTS

-name *bundle_name*

Specifies a unique name for the bundle. If this option is not specified, the command generates a bundle named BUNDLE_n, where n is the current number of existing bundles plus one.

objects

Specifies the objects to add to the bundle. If this option is not specified, the tool creates an empty bundle. Objects specified can be supernets.

DESCRIPTION

This command creates a new homogeneous bundle of nets or supernets in the current design. If you specify a name for the bundle with the **-name** option, the name must be unique. Use the **remove_bundles** command to delete a bundle.

EXAMPLES

The following examples create new bundles in the design.

```
prompt> create_bundle
{BUNDLE_5}
prompt> create_bundle -name my_bundle
```

```
{my_bundle}
prompt> create_bundle -name my_bundle_3 [get_nets abc]
{my_bundle_3}
```

SEE ALSO

add_to_bundle(2)
get_bundles(2)
remove_bundles(2)
remove_from_bundle(2)
report_bundles(2)

create_bundles_from_patterns

Groups nets into bundles based on net name and the number of nets in the net group.

SYNTAX

status **create_bundles_from_patterns**
[-net_name_prefix *prefix*]
[-net_order ascending | descending]
[-minimum_nets *min*]
[-maximum_nets *max*]
[-no_braces]
[-no_brackets]
[-no_angle_brackets]
[-no_underlines]
[-no_colons]
[-no_parentheses]
[-bundle_grouping]

Data Types

prefix string
min integer
max integer

ARGUMENTS

-net_name_prefix *prefix*

Specifies the net name prefix to match when creating bundles. By default, the command creates bundles by matching any unique net group names that contain eight or more bits.

-net_order *ascending* | *descending*

Specifies the net sorting order. By default, the order is ascending.

-minimum_nets *min*

Specifies the minimum number of nets in a bundle. By default, the tool creates bundles for net groups with eight or more nets.

-maximum_nets *max*

Specifies the maximum number of nets in a bundle. By default, there is no limit to the number of nets in a bundle.

-no_braces

Ignore net group names that contain braces ({}).

-no_brackets

Ignore net group names that contain brackets ([]).

-no_angle_brackets

Ignore net group names that contain angle brackets (<>).

-no_underlines

Ignore net group names that contain underscore characters (_).

-no_colons

Ignore net group names that contain colon characters (:).

-no_parentheses

Ignore net group names that contain parentheses (()).

-bundle_grouping

Enable bundle grouping to enable the command to create bundles of other bundles.

DESCRIPTION

This command groups nets into bundles based on net name and number of bits. You can use this command to quickly create bundles from net groups without specifying individual **create_bundle** commands. Net group names are extracted from the design by the tool, or by specifying the **-net_name_prefix** option. You can also specify the bit ordering in the bundle with the **-net_order** option. The tool extracts the net groups and determines the number of bits in the group. Only net groups with eight or more bits, or groups that meet the group size criteria set with the **-minimum_nets** and **-maximum_nets** options, are made into net bundles.

This command supports net group indexes that contain the following characters:

bus[i]
bus{i}
bus<i>
bus(i)
bus_i
bus:i

EXAMPLES

The following example creates a bundle of signals that begin with "ahbmi" and reports the contents of the bundle.

```
prompt> create_bundles_from_patterns -net_name_prefix ahbmi
Information: There are 1 new bundles defined. (LED-102)
{B_ahbmi}

prompt> report_bundles [get_bundles {B_ahbmi}]
Begin bundle report ..
  Bundle NAME: `B_ahbmi'
  Number of objects in bundle 24
```

```
  Bundle Type 'net'
  Objects in bundle
    Position = 0, net ahbmi[0]
    Position = 1, net ahbmi[1]
    Position = 2, net ahbmi[2]
    ...
End bundle report
1
```

SEE ALSO

add_to_bundle(2)
create_bundle(2)
get_bundles(2)
remove_bundles(2)
remove_from_bundle(2)
report_bundles(2)

create_bus_routing_style

Creates an analog constraint group with bus_style intent in the current design.

SYNTAX

collection **create_bus_routing_style**
[*constraint_group_name*]
-for *objects*
[-gap *distance*]
[-shield_placement default | double_interleave | half_interleave | interleave | outside]
[-corner_type cross | default | river]
[-valid_layers *layer_list*]
[-layer_spacings {*layer1 spacing1* ...}]
[-layer_widths {*layer1 width1* ...}]
[-force]

Data Types

constraint_group_name string
objects string or collection
distance float
layer_list list
layer1 string
spacing1 float
width1 float

ARGUMENTS

constraint_group_name

Specifies the name of the constraint group. By default, the command generates a unique constraint group named "bus_style_n", where n is a unique, monotonically increasing integer value.

-for *objects*

Specifies the objects - can be nets, bundles, or topology_edges - that constitute the analog constraint group. It can be specified as a string (name of objects) or the collection of objects.

-gap *distance*

Specifies the spacing between the outermost bus bits and other shapes in the block. The default is zero.

-shield_placement default | double_interleave | half_interleave | interleave | outside

Specifies the shielding type for the main bus trunk, if needed. Allowed values are *default*, *double_interleave*, *half_interleave*,

interleave and *outside*. The default is *default*.

-corner_type cross | **default** | **river**

Specifies the bus corner type. Allowed values are *cross*, *default* and *river*. The default is *default*.

-valid_layers *layers*

Specifies the list of layer names to be used for bus trunk. If specified, layers type is set to range, otherwise it is set to default. When specified the min and max layers are derived from the list.

-layer_spacings {*layer1 spacing1* ...}

Specifies the spacing of a layer used in trunk routing. This is a list of alternating layers and spacings, for example, {M1 5.0 M2 6.0}.

-layer_widths *list*

Specifies the width of a layer used in trunk routing. This is a list of alternating layers and widths, for example, {M1 5.0 M2 6.0}.

-force

Forces the deletion of an existing constraint group if the name matches the specified *constraint_group_name*. The default is false, and the command exits with an error if the name already exists.

DESCRIPTION

The **create_bus_routing_style** command creates a constraint group for a collection of nets, bundles, and topology_edges with bus_style intent. The constraint is used by Custom Router.

You can specify the valid set of layers for trunk routing, layer-specific spacings and widths. The shielding placement of bus trunk can also be specified.

EXAMPLES

The following example creates a constraint group with system generated name "bus_style_1" for nets matching pattern "pr*" which will be of bus_style intent. The shield placement is interleave. Also a spacing of 3 will be used for layer M2 and spacing of 5 will be used for layer M3. The 'pr*' nets are routed with Custom Router.

```
prompt> create_bus_routing_style -for pr* -shield_placement interleave \
-layer_spacings {M2 3 M3 5}
{bus_style_1}
```

SEE ALSO

get_constraint_groups(2)
remove_constraint_groups(2)
report_constraint_groups(2)

create_busplans

Creates one or more busplan objects. Busplan objects are used during bus planning.

SYNTAX

int **create_busplans**
[-name *name*]
[-rule *net_estimation_rule*]
[-auto]
[-from *pins_or_ports*]
[-to *pins_or_ports*]
[-nets *nets*]
[-force]
[-add_start_end_cells *cells*]
[-verbose]
[-reset]

Data Types

name string
net_estimation_rule string
pins_or_ports collection
nets collection
cells collection

ARGUMENTS

-name *name*

Specify the name of the busplan object. If you do not specify a name, a unique name of the form bus* will be generated automatically.

-rule *net_estimation_rule*

Specify a net_estimation_rule, which must be previously declared by using the **set_net_estimation_rule** command. If you do not define a rule, the "default" rule is used. The rule is used when calculating the delay of segments in the budget. In particular, the "ns_per_mm" parameter of the rule will be multiplied by the physical length of the segment to get a delay value.

-auto

Specify that the tool should automatically find buses in the design. You must specify one of *-auto*, *-from* or *-nets*.

-from *pins_or_ports*

Specify where the busplan should begin. If you specify ports, they must be input ports. If you specify pins, they must be output

pins of blocks.

You must specify one of *-auto*, *-from* or *-nets*.

-to *pins_or_ports*

Specify where the busplan should end. If you specify ports, they must be output ports. If you specify pins, they must be input pins of blocks.

-nets *nets*

Specify a set of nets through which the busplan should pass. The net must be in the direct fanout of a legal bus startpoint, separated only by buffers or inverters. See the documentation of the *-from* option for a definition of legal bus start points.

You must specify one of *-auto*, *-from* or *-nets*.

-force

Create the busplan, even if it crosses another busplan. If you use this option, then some or all of the existing busplans are removed and replaced by the new busplan definition. By default, a busplan will not be defined in a place where another busplan already exists.

-add_start_end_cells *cells*

Specifies to add to the list of cell instances where buses can start or end. By default it will contain all top level hard macros and blocks. This information is stored by the command, you do not need to use **-add_start_end_cells** on the next command invocation.

-verbose

Output more information when running.

-reset

Remove all busplan information from design.

DESCRIPTION

This command creates a database object called a "busplan", that can be used to do register planning. A busplan is a collection of paths through the design that start at top-level ports or block outputs and end at top-level ports, or block inputs. The paths may cross through multiple levels of hierarchy and may also traverse through the following circuit elements:

- buffers and inverters
- level shifters
- isolation cells
- flip-flops and latches

Busplans are declared by specifying either *-from* or *-nets* to indicate where the bus starts or the tool can automatically find and create busplans using *-auto*. You can specify many start pins, ports or nets at the same time, to form a multibit busplan.

The busplan also has a net_estimation_rule object associated with it, which you set by using the *-rule* option. The net_estimation_rule object must have been previously declared by using the **set_net_estimation_rule** command. The rule will be used when calculating the delay of segments in the busplan. In particular, the "ns_per_mm" parameter of the rule will be multiplied by the physical length of the segment to get a delay value.

This command returns 1 on success, 0 otherwise.

EXAMPLES

Create a net estimation rule with 0.5 ns_per_mm propagation time. Define 8 bit busplan starting at block B1 and ending at block B2. Report on the segments along the busplan.

```
prompt> set_net_estimation_rule fast_rule -parameter ns_per_mm -value 0.5
prompt> create_busplans -name bus1 -rule fast_rule -from B1/OUT1[*] -to B2/IN1[*]
prompt> report_busplans -start_end_cells bus1

****************************************
Report : report_busplans
Design : TOP
Version: L-2016.03-DEV
Date   : Sun Aug  2 16:55:49 2015
****************************************
Start/end at instances: B1 B2 B3 B4
bus1: 8 bits start=B1/OUT1[7] clock=-unknown-
  pin              group1     (  8) B1/OUT1[7]
    register         group2     (  8) r1/r7
      register       group3     (  8) r2/r7
        pin          group4     (  8) B2/IN1[7]
1
```

SEE ALSO

report_busplans(2)
remove_busplans(2)
set_net_estimation_rule(2)
report_net_estimation_rules(2)

create_cell

Creates one or more cells referencing a library cell, module, or via_def

SYNTAX

collection **create_cell**
[-design *design*]
cell_names
reference

Data Types

design collection
cell_names list
reference collection

ARGUMENTS

-design *design*

Specifies the design in which to create the cells. If no design is specified, the cells are created in the current design.

cell_names

Specifies the names of the cells to be created in the design. Each cell name must be unique.

reference

Specifies the name of the library cell or module the created cells should instantiate, the reference must already exist.

The reference may also be a via_def when creating through-silicon via ("TSV") cells, in which case the via_def must be of type "through_silicon_via_def". TSV cells, which are identified by their *is_tsv* attribute being true, are created by this command if the reference is either a TSV via_def or a design with design_type "tsv". If a TSV design, it must meet certain criteria, which as of this writing are:

- Its *design_type* attribute must be "tsv",
- It must have exactly one port with two terminals,
- Its *boundary* attribute must be set.

DESCRIPTION

This command creates new cells in the current design (unless otherwise specified by -design). One cell is created for each of the

names listed.

EXAMPLES

The following example creates three instances of the library cell *mylib/ND2*.

```
prompt> create_cell {U23 U1/U24 U2/U7/U25} myLib/ND2


{U23 U1/U24 U2/U7/U25}
```

The following example creates three instances of the library cell *ND2*. It will work similer to **/ND2*

```
prompt> create_cell {U23 U1/U24 U2/U7/U25} ND2


{U23 U1/U24 U2/U7/U25}
```

The following example creates two instances of the module MID.

```
prompt> create_cell {U110 U111} [get_modules MID]


{U110 U111}
```

The following example creates an instance of the lib cell myLib/ND4 using **get_blocks** to find the reference. **get_blocks -all** is needed if the lib_cell is not open already.

```
prompt> create_cell U210 [get_blocks -all myLib:ND4.design]


{U110 U111}
```

SEE ALSO

create_module(2)
create_net(2)
connect_net(2)
edit_module(2)
get_lib_cells(2)
get_modules(2)
get_blocks(2)
remove_cells(2)

create_cell_array

Inserts cells in the block in an array style.

SYNTAX

status **create_cell_array**
-lib_cell *lib_cell*
[-voltage_area *voltage_area*]
[-boundary *polygon*]
[-x_pitch *distance*]
[-y_pitch *distance*]
[-x_offset *distance*]
[-y_offset *distance*]
[-orient R0|R90|R180|R270|MX|MY|MXR90|MYR90]
[-checkerboard even | odd]
[-snap_to_site_row true | false]
[-preserve_boundary_row_lib_cells *lib_cell_list*]
[-prefix *prefix*]

Data Types

lib_cell string
voltage_area collection
polygon list
distance float
lib_cell_list collection
prefix string

ARGUMENTS

-lib_cell *lib_cell*

Specifies the library cell to insert. You must specify a single library cell. This is a required option.

-voltage_area *voltage_area*

Specifies the voltage area in which to insert the cells.

If you do not specify this option or the **-boundary** option, the insertion region is the core area.

If you specify both the **-voltage_area** and **-boundary** options, the insertion region is the overlapping region between the specified voltage area and polygon.

-boundary *polygon*

Specifies the region in which to insert the cells.

If you do not specify this option or the **-voltage_area** option, the insertion region is the core area.

If you specify both the **-voltage_area** and **-boundary** options, the insertion region is the overlapping region between the specified voltage area and polygon.

-x_pitch ***distance***

Specifies the horizontal distance between the centers of two cells.

If you do not specify this option, the command inserts a single column of cells whose spacing is specified by the **-y_pitch** option.

If you do not specify this option or the **-y_pitch** option, the command inserts a single cell at the specified offset location.

-y_pitch ***distance***

Specifies the vertical distance between the centers of two cells.

If you do not specify this option, the command inserts a single row of cells whose spacing is specified by the **-x_pitch** option.

If you do not specify this option or the **-x_pitch** option, the command inserts a single cell at the specified offset location.

-x_offset ***distance***

Specifies the horizontal offset from the insertion region boundary.

The default is 0.

-y_offset ***distance***

Specifies the vertical offset from the insertion region boundary.

The default is 0.

-orient R0|R90|R180|R270|MX|MY|MXR90|MYR90

Specifies the orientation of the inserted cells. Valid values are **R0**, **R90**, **R180**, **R270**, **MX**, **MY**, **MXR90**, and **MYR90**.

If you do not specify this option, the default orientation depends on the setting of the **-snap_to_site_row** option.

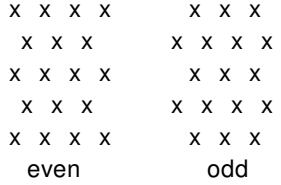
- If **true**, the default orientation is the same as the site row orientation.
- If **false**, the default orientation is **R0**.

-checkerboard even | **odd**

Inserts cells in a checkerboard fashion. There are two types of checkerboard: **even** or **odd**.

- The even checkerboard type specifies that the first cell is placed at the lower-left corner of the insertion region.
- The odd checkerboard type specifies that a blank space is placed at the lower-left corner of the insertion region.

The following figure shows the checkerboard insertion patterns:



If you do not specify this option, the command inserts the cells as a normal array.

-snap_to_site_row true | **false**

Specifies if the new cells are snapped to site rows.

The default is **true**.

-preserve_boundary_row_lib_cells ***lib_cell_list***

Specifies the library cells, such as critical area breaker cells or boundary cells, that require an abutting tap cell. If an empty space in the checkerboard pattern occurs above or below an instance of one of the specified library cells, the command inserts a tap cell in that empty space.

This option is valid only when used with the **-checkerboard** option.

-prefix ***prefix***

Specifies the prefix added to the inserted cell names.

When you use this option, the command uses the following naming convention for the inserted cells:

<*prefix*>__<*lib_cell*>_R#_C#_*number*

By default, no prefix is added and the tool uses the following naming convention for the inserted cells:

headerfooter__<*lib_cell*>_R#_C#_*number*

DESCRIPTION

The **create_cell_array** command inserts an array of cells into the block.

To insert power-switch cells, use the **create_power_switch_array** command instead of this command.

EXAMPLES

The following example inserts an array of HEAD_16DM cells in the current block.

```
prompt> create_cell_array -lib_cell HEAD_16DM \
  -x_pitch 10 -y_pitch 10
Insert 684 cell HEAD16DM successfully.
```

SEE ALSO

create_power_switch_array(2)
create_tap_cells(2)

create_cell_array_pattern

Creates a cell array pattern in the current or specified block.

SYNTAX

collection **create_cell_array_pattern**
[-name *name*]
[-cell *cell*]
[-type solid | checkerboard]
[-lib_cell_name *name*]
[-row_column_ratio *ratio*]
[-horizontal_spacing *spacing*]
[-vertical_spacing *spacing*]
[-halo *factor*]

Data Types

name string
cell collection
ratio int_pair
spacing integer
factor float

ARGUMENTS

-name *name*

Specifies the name of the cell_array_pattern. The name may not contain the special characters '~' or '/'. If unspecified, the command will generate a name automatically using the prefix "CELL_ARRAY_PATTERN" with a system-generated number appended.

-cell *cell*

Specifies that this cell_array_pattern should be created in the block referenced by *cell*.

-type solid | checkerboard

Specifies the type of the cell_array_pattern. "solid" means that an instance of the lib_cell should be in every eligible site within the pattern. "checkerboard" means that an instance of the lib_cell should be in every other eligible site along each row, with the rows staggered such that an instance of the lib_cell is in every other eligible site along the columns as well.

-lib_cell_name *name*

Specifies the name of the lib_cell to be instantiated within the pattern.

-row_column_ratio *ratio*

Specifies the ratio of rows to columns in the cell_array_pattern. The ratio is an integer pair, with each integer >= 1. Note that the actual number of rows and columns can be any positive integer multiple of these values.

-horizontal_spacing *spacing*

Specifies the number of sites along each row of the cell_array_pattern between each eligible site for lib_cell placement within the pattern.

-vertical_spacing *spacing*

Specifies the number of sites along each column of the cell_array_pattern between each eligible site for lib_cell placement within the pattern.

-halo *factor*

Specifies the halo of the cell_array_pattern. The *factor* is the multiplier of the cell_array_patterns width and height, which indicates the width of the halo in the respective horizontal and vertical directions.

DESCRIPTION

The **create_cell_array_pattern** command creates a new cell_array_pattern in the current block, or the block specified by the *-cell* argument. A cell_array_pattern stores a description of how to place lib_cells associated with topological objects during the interconnect planning phase.

The name of a cell_array_pattern must be unique within its block.

Here is an illustration of two types of cell_array_patterns, each with row_column_ratio of {2 3} and horizontal and vertical spacing of 1:

type = solid, number of rows = 2

[*][ ][*][ ][*] [ ][ ][ ][ ][ ] [*][ ][*][ ][*]

type = checkerboard, number of rows = 4

[*][ ][ ][ ][*][ ][ ][ ][*][ ][ ] [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][*][ ][ ][ ][*][ ][ ][ ][*] [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ] [*][ ][ ][ ][*][ ][ ][ ][*][ ][ ] [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ] [ ][ ][*][ ][ ][ ][*][ ][ ][ ][*]

EXAMPLES

The following example creates a cell_array_pattern in the current block:

```
prompt> create_cell_array_pattern
{CELL_ARRAY_PATTERN0}
```

The following example creates a cell_array_pattern with name "mycap", type "checkerboard", row/column ratio of 2/3, and a halo that is 10% of the dimensions of the pattern:

```
prompt> create_cell_array_pattern -name mycap -type checkerboard -row_column_ratio {2 3} -halo 0.1
{mycap}
```

SEE ALSO

get_cell_array_patterns(2)
remove_cell_array_patterns(2)
report_cell_array_patterns(2)

create_cell_bus

Creates one cell bus and it cell members referencing a library cell or module.

SYNTAX

collection **create_cell_bus**
[-design *design*]
[-create_cells]
[-block *block*]
[-cell *cell*]
-reference *module*
cell_names

Data Types

design collection
block collection
cell collection
cell_bus_name string

ARGUMENTS

-design *design*

Specifies the top-level design for finding objects. If this is not specified, objects are found in the current design.

-create_cells

Creates cell bus members if they do not exist.

-block *block*

Specifies the block where the cell bus is to be added. If specified, **create_cell_bus** has the same effect as **edit_module** in which the module is changed and changes are applied to all module occurrences.

-cell *cell*

Specifies the cell where the cell bus is to be added. The cell bus is created on the cell's reference module. **create_cell_bus** first uniquifies the reference if needed, then creates the cell bus. When not specified, the cell bus is created on the current module.

-reference *module*

Specifies the name of the library cell or module the created cell_bus's member should instantiate, the reference must already exist.

cell_bus_name

Specifies the cell bus name and its range using the **name [start: end]** format. Specify a cell bus named busA with cell from 1 *to* 5 as **busA[1:5]**. Specify a cell bus named busB with cell from 5 *downto* as **busB [5:1]**.

DESCRIPTION

This command creates a cell bus, collects or creates cell bus members, and puts them into the cell bus created.

The command returns the created cell_bus(as a collection), an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

EXAMPLES

The following example creates cell bus named *bus1* from cell 1 to 5 as bus[1], bus[2], bus[3], bus[4], bus[5] whose reference is RISC_CORE.

```
prompt> create_cell_bus -reference RISC_CORE bus1[1:5]
{"bus1"}
```

The following example first creates cells with reference as RISC_CORE, then creates a cell bus named *bus2* from the created cell with indexes 5 downto 1 as bus2[5], bus2[4], bus2[3], bus2[2], bus2[1]

```
prompt> create_cell_bus -reference RISC_CORE bus2[5:1] -create_cells
{"bus2"}
```

SEE ALSO

get_cell_buses(2)
remove_cell_buses(2)

create_channel_congestion_map

Creates a congestion map for the channel areas of the design.

SYNTAX

int **create_channel_congestion_map**
[-channel_width_threshold *threshold*]
[-boundary *coordinates*]
[-congestion_map_only true | false]

Data Types

threshold	float
coordinates	list

ARGUMENTS

-channel_width_threshold *threshold*

Specifies the threshold for the channel width in microns. Only areas with a width less than or equal to the threshold are considered a channel. By default, if you omit both the **-channel_width_threshold** and **-boundary** options, the tool defines the channel width as 10% of the longest chip dimension and generates a congestion map for the entire design.

-congestion_map_only true | false

Creates the congestion map without creating global route segments (glinks). By default, this option is false and the command generates glinks. If design has existing glinks, the command will not remove them if this option is true and the new congestion map could potentially be out of sync comparing with the existing glinks.

-boundary *coordinates*

Specifies the area in which to generate the considered map. The areas are boundaries specified by a list of rectangles or polygons, with units in microns. Specify the coordinates by using one of the following formats:

-boundary {{*llx lly*} {*urx ury*}}
-boundary {{*x1 y1*} {*x2 y2*} {*x3 y3*} ...}

In the first format, *llx* and *lly* specify the lower-left x- and y-coordinates, and *urx* and *ury* specify the upper-right x- and y-coordinates. In the second format, you specify each corner on the polygon to define the boundary.

When used together with **-channel_width_threshold** option, the command routes channel areas defined by either the **-channel_width_threshold** or **-boundary** options. By default, if you omit both **-channel_width_threshold** and **-boundary** options, the tool defines the channel width as 10% of the longest chip dimension and generates a congestion map for those channels in the entire design.

DESCRIPTION

This command creates a global route-based congestion map for the specified channel or boundary areas. An area is considered a channel if it is enclosed by macro cell boundaries, die boundaries, or block boundaries.

Since this command uses global routing to generate a congestion map, it is possible that congestion might appear in areas outside of the channel areas.

Since this command uses fast, very low effort global routing and other simplifying assumptions to generate a congestion map quickly, it will produce results that are more pessimistic than a congestion map generated by the **route_global -floorplan true** command. Similarly, the results from the **route_global -floorplan true** command will be slightly more pessimistic than results from **route_global**, because **route_global -floorplan true** performs fast, lower effort global routing along with simplifying assumptions to complete global routing quickly.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following example creates a channel congestion map for channel areas with a width less than 10 microns.

```
prompt> create_channel_congestion_map -channel_width_threshold 10
```

SEE ALSO

route_global(2)

create_check_design_strategy

Defines checks which can be run through **check_design** command.

SYNTAX

status **create_check_design_strategy**
[-define_check *atomic_check_name*]
[-define_group *mega_check_name*]
definition

Data Types

atomic_check_name	string
mega_check_name	string
definition	list

ARGUMENTS

-define_check *atomic_check_name*

Specifies the name of the atomic check to create. See **check_design** for definition of atomic-check.

If the specified name is the name of a predefined check (built-in check), the command issues an error message.

If the specified name is the name of an existing user-defined check, the new definition overwrites the previous definition.

The **-define_check** and **-define_group** options are mutually exclusive.

-define_group *mega_check_name*

Specifies the name of the mega check to create. See **check_design** for definition of mega-check.

In place of **mega_check_definition**, user should provide a list of atomic-check names. These atomic-check names can be names of any user-defined or pre-defined atomic-checks.

If the specified name is the name of a predefined check (built-in check), the command issues an error message.

If the specified name is the name of an existing user-defined check, the new definition overwrites the previous definition.

The **-define_check** and **-define_group** options are mutually exclusive.

definition

Defines the check.

When you use the **-define_check** option, this argument defines the atomic check. You must specify a single command with or without its options.

When you use the **-define_group** option, this argument defines the mega check. You must specify a list of atomic-check names. These atomic-check names can be the names of any user-defined or predefined atomic checks.

DESCRIPTION

User can use this command to,

(1) Define a new atomic-check or mega-check, whose name does not match any of the existing pre-defined or user-defined checks. For example, user can define a new check by name say, **my_check** and none of the pre-defined checks or existing user-defined checks have the name **my_check**.

(2) Re-define existing user-defined checks. For example, lets say a user-defined check by name **my_pre_placement_stage** already exists. User can give a new definition to the check **my_pre_placement_stage**, which overrides the previous definition.

To report the checks defined or modified by the **create_check_design_strategy** command, use the **report_check_design_strategy** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

(1) The following example defines a new user-defined check **my_mv_design**,

```
prompt> create_check_design_strategy -define_check my_mv_design \
   {check_mv_design -power_connectivity -max_message_count 0}
```

(2) The following example defines a mega-check named **mv_cts** using the existing atomic-checks named mv_design, cts_qor, and legality:

```
prompt> create_check_design_strategy -define_group mv_cts \
   {mv_design cts_qor legality}
```

SEE ALSO

check_design(2)
report_check_design_strategy(2)
get_design_checks(2)

create_checkpoint_action

Defines an action in the checkpoint system that can then be associated to run at a checkpoint.

SYNTAX

string **create_checkpoint_action**
action_name
tcl_commands

Data Types

action_name string
tcl_commands list

ARGUMENTS

action_name

Specifies the name of the action or flow change for the checkpoint system. A mandatory option specifying the name of the flow change. The name of the flow change must be unique, otherwise it will overwrite the previous definition.

tcl_commands

Specifies the block of TCL commands to be associated with the given action or flow change for the checkpoint system.

DESCRIPTION

Defines an action with the given name in the checkpoint system. An action can be any block of one or more Tcl commands. This command defines an action, but it does not execute the action. For an action to run, it must be associated to one or more checkpoints using the associate_checkpoint_action command. An action can be associated to run before, after, or as a replacement for the checkpoint contents.

Actions are used to modify behavior of the default flow. They allow experimental flow changes to be defined independently of the golden flow scripts. They may be defined in the ./checkpoint.config.tcl file in the run directory, which will automatically be sourced by the checkpoint system.

The simplest and most common example is an action which applies an app option setting before a checkpoint. But more complex actions are possible, such as applying multiple tool constraints, or modifying the options on a checkpointed command.

Actions execute at the same script scope as the checkpoint to which they are associated. So, any local variables or other variables made available through the global or upvar commands can be used inside that action. For this reason, ensure that any variables

you define inside an action do not conflict with variables defined at the scope in which it will run.

EXAMPLES

Defines an action called postroute_ccd that activates the CCD route_opt app option. Then action is then associated to run before the first_route_opt checkpoint. When the first_route_opt checkpoint is executed in the block's flow script, this action will execute beforehand.

```
## checkpoint.config.tcl
create_checkpoint_action postroute_ccd {
        set_app_options -name route_opt.flow.enable_ccd -value true
}

associate_checkpoint_action -enable postroute_ccd -before first_route_opt

## Flow route_opt.tcl script
eval_checkpoint first_route_opt { route_opt }
```

Defines and associates an action to replace the existing create_placement call at the incremental_place checkpoint with an option to force high congestion effort. The get_current_checkpoint -command option is used to access the current checkpointed command contents and modify them with the new option.

```
## checkpoint.config.tcl
create_checkpoint_action high_effort_congestion {
        eval [get_current_checkpoint -command] -congestion -congestion_effort high

}

associate_checkpoint_action -enable high_effort_congestion \
        -replace incremental_place

## Flow place_opt.tcl script
eval_checkpoint incremental_place {
        create_placement -incremental -timing_driven
}
```

SEE ALSO

remove_checkpoint_actions(2)
create_checkpoint_report(2)
remove_checkpoint_reports(2)
associate_checkpoint_action(2)
associate_checkpoint_report(2)
eval_checkpoint(2)
get_current_checkpoint(2)
set_checkpoint_options(2)
reset_checkpoints(2)

get_checkpoint_data(2)

create_checkpoint_report

Defines a report in the checkpoint system that can then be associated to run at a checkpoint.

SYNTAX

string **create_checkpoint_report**
report_name
tcl_commands

Data Types

report_name string
tcl_commands list

ARGUMENTS

report_name

Specifies the name of the report for the checkpoint system. A mandatory option specifying the name of the report. The name of the report must be unique, otherwise it will overwrite the previous definition.

tcl_commands

Specifies the block of TCL commands to be associated with the given report for the checkpoint system.

DESCRIPTION

Defines a report with the given name in the checkpoint system. A report can be any block of one or more Tcl commands. This command defines a report, but it does not execute the report. For a report to run, it must be associated to one or more checkpoints using the associate_checkpoint_report command. A report can be associated to run before or after a checkpoint. Unlike actions, reports cannot replace a checkpoint's contents.

Reports are used to generate output from the tool at a checkpoint. They allow report definitions to be defined independently from the golden flow scripts. Each user may apply their preferred reporting setup to write out the required data needed for debugging run results. Typically, reports are defined in the ./checkpoint.config.tcl file in the run directory, which will automatically be sourced by the checkpoint system.

Checkpoint reports are typically used to write out common reports like report_qor, report_timing, check_mv_design, check_routes, other default tool reports, or custom user reports.

Reports execute at their own scope. They do not have access to the variables at the scope of the associated checkpoint, unless they

are made available in the create_checkpoint_report body with the Tcl global or upvar command. Any variables you define in the body of create_checkpoint_report will not conflict with variables at the checkpoint scope.

EXAMPLES

This example defines a report named timing that generates common timing related reports into the ./checkpoint directory. The get_current_checkpoint -name command is used to get the name of the current checkpoint, which is used to give each written report a unique and meaningful name.

This report is then associated to run after all *-final checkpoint names. When the flow script runs, any eval_checkpoint commands with checkpoint names matching *-final will run this report.

```
## checkpoint.config.tcl
create_checkpoint_report timing {

set name [get_current_checkpoint -name]
redirect -file ./checkpoint/${name}.qor.rpt { report_qor }
redirect -append ./checkpoint/${name}.qor.rpt { report_qor -summary }
redirect -file ./checkpoint/${name}.timing.rpt { report_timing }

}
associate_checkpoint_report -enable timing -after *-final
```

This example defines a report that saves a block to the NDM design database. Since it can be useful to save a before or after an important flow step, the report uses the get_current_checkpoint -position option to uniquely name the saved block, indicating whether it was saved 'before' or 'after' the checkpoint. This report can then be associated to run before or after any checkpoint names where the user wants to save the database.

```
## checkpoint.config.tcl
create_checkpoint_report save_block {

set name [get_current_checkpoint -name]
set position [get_current_checkpoint -position]
save_block -as checkpoint_${position}_${name}

}
```

SEE ALSO

remove_checkpoint_reports(2)
create_checkpoint_action(2)
remove_checkpoint_actions(2)
associate_checkpoint_action(2)
associate_checkpoint_report(2)
eval_checkpoint(2)
get_current_checkpoint(2)
set_checkpoint_options(2)

reset_checkpoints(2)
get_checkpoint_data(2)

create_clock

Creates a clock object.

SYNTAX

collection **create_clock**
-period *period_value*
[-name *clock_name*]
[-waveform *edge_list*]
[-add]
[*source_objects*]
[-comment *comment*]

Data Types

period_value float
clock_name string
edge_list list
source_objects list
comment string

ARGUMENTS

-period *period_value*

Specifies the clock period in library time units. This is the minimum time over which the clock waveform repeats. The *period_value* must be greater than or equal to zero.

-name *clock_name*

Specifies the name of the clock being created, enclosed in quotation marks. If you do not use this option, the clock gets the same name as the first clock source specified in the *sources* option. If you did not specify *sources*, you must use the **-name** option, which creates a virtual clock not associated with a port or pin. You can use both the **-name** option and the *sources* option to give the clock a more descriptive name than the first source pin or port. If you specify the *-add* option, you must use the **-name** option and the clocks with the same source must have different names.

-waveform *edge_list*

Specifies the rise and fall edge times of the clock waveforms of the clock, in library time units, over an entire clock period. The first time in the *edge_list* is a rising transition, typically the first rising transition after time zero. There must be an even number of edges, and they are assumed to be alternating rise and fall. The edges must be monotonically increasing, except for a special case with two edges, where fall can be less than rise. The numbers should represent one full clock period. If you do not specify this option, a default waveform is assumed, which has a rise edge of 0.0 and a fall edge of *period_value*/2.

-add

Specifies whether to add this clock to the existing clock or to overwrite. Use this option to capture the case where multiple clocks must be specified on the same source for simultaneous analysis with different clock waveforms. When you specify this option, you must also use the *-name* option.

Defining multiple clocks on the same source pin or port causes longer runtime and higher memory usage than a single clock, because the tool must explore all possible combinations of launch and capture clocks. Use the **set_false_path** command to disable unwanted clock combinations.

source_objects

Specifies the objects used as sources of the clock. The sources can be ports, pins or nets in the design. If you do not use this option, you must use the **-name** option, which creates a virtual clock not associated with a port, pin or net. If you specify a clock on a pin that already has a clock, the new clock replaces the old one unless you use the **-add** option. When a net is used as the source, the first driver pin of the net is the actual source used in creating the clock.

-comment ***comment***

Associate a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

DESCRIPTION

Creates a clock object. It is created in the current design and is applied to the specified *sources*.

If successful, the command returns a collection containing the new or modified clock.

If you do not specify a *sources* value, but give a *clock_name* value, a virtual clock is created. You can create a virtual clock to represent an off-chip clock for input or output delay specification. For more information about input and output delay, see the **set_input_delay** and **set_output_delay** man pages.

If one of the *sources* is already the source of a clock, the source is removed from that clock. This clock is eliminated if it has just one source.

The **create_clock** command also defines the waveform for the clock. The clock can have multiple pulses per period.

The **create_clock** command is used together with the **set_clock_latency**, **set_clock_uncertainty**, **set_propagated_clock**, and **set_clock_transition** commands to specify properties of clock networks. By default, a new path group is created for the clock. This new path group brings together the endpoints related to this clock for cost function calculation. To remove the clock from its assigned group, use the **group_path** command to reassign the clock to another group or to the default path group. For more information, see the **group_path** man page.

The new clock has ideal clock latency and transition time; no propagated delay through the clock network is assumed and a transition time of zero is used at the clock source pin. To enable propagated latency for a clock network, use the **set_propagated_clock** command. To set an estimated latency, use the **set_clock_latency** command.

To show information about clocks in the design, use the **report_clock** command. To create a collection of clocks matching a pattern and optionally matching filter criteria, use the **get_clocks** command.

To undo **create_clock**, use the **remove_clock** command.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example creates a clock on a port named *PHI1* with a period of *10.0*, a rise at *5.0*, and a fall at *9.5*.

```
prompt> create_clock PHI1 -period 10 -waveform { 5.0 9.5 }
```

The following example shows a clock named *PHI2* with a falling edge at *5* and a rising edge at *10* with a period of *10*. Because the edges for the **-waveform** option should be ordered as first rise, then fall, and should increase in value, the fall edge can be given as *15*. This places the next falling edge after the first rise edge at *10*.

```
prompt> create_clock PHI2 -period 10 -waveform { 10 15 }
```

The following example creates a virtual clock *PHI2* with a period of *10.0*, a rise at *0.0*, and a fall at *5.0*.

```
prompt> create_clock -name PHI2 -period 10 -waveform {0.0 5.0}
```

The following example creates a clock named *clk2* with multiple sources.

```
prompt> create_clock -name clk2 -period 10 \
  -waveform {2.0 4.0} {clkgen1/Z clkgen2/Z clkgen3/Z}
```

The following example creates a clock named *CLK* on pin *u13/Z* with a period of *25*, a fall at *0.0*, a rise at *5.0*, a fall at *10.0*, a rise at *15.0*, and so on.

```
prompt> create_clock u13/Z -name CLK -period 25 -waveform { 5 10 15 25}
```

SEE ALSO

all_clocks(2)
get_clocks(2)
group_path(2)
remove_clocks(2)
report_clocks(2)
set_clock_latency(2)
set_clock_uncertainty(2)
set_input_delay(2)
set_output_delay(2)

create_clock_balance_group

Creates a clock balance group containing clocks or skew groups that need to be balanced together.

SYNTAX

string **create_clock_balance_group**
-objects *object_list*
-name *group_name*
[-offset_latencies *offset_value_list*]
[-corner *corner_name*]

Data Types

object_list	collection
group_name	string
offset_value_list	list
corner_name	string

ARGUMENTS

-objects *object_list*

Specifies the list of clocks or skew groups to be balanced together. This list cannot contain a combination of clocks and skew groups. Note that objects list should contain at least two objects.

-name *group_name*

Specifies the name of the clock balance group.

-offset_latencies *offset_value_list*

Specifies the ordered list of offset latency values for the clocks in '-objects' argument. The default value is 0.0 for all objects in the objects list.

-corner *corner_name*

Specifies the corner to which the offset latencies are applied.

DESCRIPTION

This command creates a new clock balance group in which all clocks have the similar insertion delay or differences by offset values if '-offset_latencies' is specified. Clocks will be balanced by using the **balance_clock_groups** command.

(a) If no offset latencies are specified, balance_clock_groups takes the longest insertion delay among the objects as the target latency and balances with 0 offset. (b) If offset latency values are set, it balances the clocks and skew groups to achieve the desired insertion delay among them with corresponding offset latencies applied.

Multicorner-Multimode Support

EXAMPLES

The following example creates a new clock balance group named 'balance_group1'.

```
prompt> create_clock_balance_group -objects {clk1 clk2 clk3} \
    -name balance_group1 \
    -offset_latencies { 0.0 0.1 -0.5 }
```

SEE ALSO

report_clock_balance_groups(2)
remove_clock_balance_groups(2)
balance_clock_groups(2)
get_clock_balance_groups(2)

create_clock_buffer

Inserts a buffer in the clock tree.

SYNTAX

string **create_clock_buffer**
[-clock *clock_name*]
[-sinks *pins_or_ports*]
[-source *pin_or_port*]
[-replace *cell*]
[-new_net_name *net_name*]
[-new_cell_name *cell_name*]
[-location {*x y*}]
[-snap true | false]

Data Types

clock_name string
pins_or_ports collection
pin_or_port string
cell string
net_name string
cell_name string
x float
y float

ARGUMENTS

-clock *clocks*

Selects the buffer based on clock-specific buffer selection rules for this clock. By default, no clock-specific buffer selection rules are considered.

-sinks *pins_or_ports*

Specifies the pins or ports to buffer. The newly inserted buffer becomes the driver of these sinks. The specified pins or ports must be driven from the same physical net, and must all be within the same physical hierarchy block. The logical nets might be different.

Exactly one of the options **-sinks**, **-source** or **-replace** must be specified.

-source *pins_or_port*

The pin or port behind which the buffer will be inserted. The newly inserted buffer becomes a sink of this source.

Exactly one of the options **-sinks**, **-source** or **-replace** must be specified.

-replace *cell*

Replaces an existing buffer with the specified new buffer. The new buffer will be connected in the same way as the old buffer. This option is used to ensure that the buffer honors the CTS buffer selection rules.

Exactly one of the options **-sinks**, **-source** or **-replace** must be specified.

-new_net_name *net_name*

Specifies a prefix to use for the new net that connects the driver of the sinks to any sinks of that driver which are not driven by the new buffer. Use this option only with the **-sinks** option, since otherwise there will be no such remaining sinks.

-new_cell_name *cell_name*

Specifies the prefix to use for the new buffer. A number will be added at the end to make the name unique. By default, the prefix is "cts_guide".

-location {*x y*}

Specifies the location to use for the new buffer. The actual location is determined by finding the nearest point not blocked by hard placement blockages or blocks. If there is no top-level free space (as in a fully-abutted design), overlap with a block is allowed.

-snap true | **false**

Specifies whether the buffer should snap to the nearest point that is not covered by a block. By default, snapping is performed, and the tool attempts to find a location at the top-level near to the indicated location to place the buffer. However, if the intention is to push the buffer down into some block, snapping can be disabled with the **-snap false** option. Note that if no top-level area is available (as in a fully-abutted design), then snapping is disabled regardless. Also note that the tool always snaps away from hard placement blockages.

DESCRIPTION

The **create_clock_buffer** command inserts a buffer in front of the specified sets of sinks, using the same rules for buffer selection as the **synthesize_clock_trees** command.

Use this command to insert guide buffers in the clock tree before clock tree synthesis to guide the clock tree synthesis.

The command returns the name of the newly inserted buffer.

EXAMPLES

Insert a buffer in front of a single sink.

```
prompt> create_clock_buffer -clock CLK -sinks [get_pins cell1/A]
cell1/cts_guide1
```

SEE ALSO

synthesize_clock_trees(2)

create_clock_drivers

Creates a set of clock driver cells in the netlist. The clock driver cells are legalized near user-specified locations. The created cells are either buffer cells, or clones of an existing template cell in the netlist. The cells can also be created inside physical hierarchy blocks.

SYNTAX

status **create_clock_drivers**
-loads *net_pin_port_list*
[-prefix *string*]
[-keepouts *poly_rect_list*]
[-max_displacement *distance*]
[-locations *coordinate_list*]
[-boxes *integer_pair*]
[-boundary *poly_rect*]
[-lib_cells *lib_cell_list*]
[-template]
[-input_pin *hier_in_pin_list*]
[-hierarchy *hier_list_per_block*]
[-short_outputs]
[-output_net_name *string*]
[-transfer_wires_from *net*]
[-configuration *config_list*]

Data Types

net_pin_port_list list or collection of a net or pins and ports
coordinate {x_coordinate y_coordinate}, in user units
coordinate_list list of coordinates
net a logical net in the block
distance distance in user units
poly_rect a polygon definition (see create_poly_rect for format)
poly_rect_list list of poly_rect
lib_cell_list list of lib_cells
integer_pair list of two positive integers {columns rows}
hier_in_pin_list list of hierarchy and input_pin pairs
hier_list_per_block list of physical hierarchy and corresponding logical hierarchy pairs
string a string value

ARGUMENTS

-loads *net_pin_port_list*

The outputs of the inserted cells will drive the loads as specified with the **-loads** option. The **-loads** option takes a single net or a

set of load pins and/or ports. If a net is specified, all load pins/ports of that (physical) net form the load pins of the inserted cells. If load pins or ports are specified explicitly, then all the specified pins and ports should initially be connected to the same (physical) net. However, not all sink pins of a net may be specified as load pins. The unspecified sink pins of the net will not be driven by the inserted cells, but will remain connected to the original net. If **-template** option is specified in the Multi-Level Physical Hierarchy flow, a list comprising the output net of the template cell in each physical hierarchy must be specified using this option.

-prefix ***string***

All newly created cells will get this prefix in front of their name. The default value is **clk_drv**. This same prefix is also applied to the names of new nets or ports that are created for a clock driver.

-keepouts ***poly_rect_list***

Specifies a list of poly_rects where no driver cells should be placed. If a preferred location is in one of the poly_rects, then no driver will be placed at that location.

-max_displacement ***distance***

The inserted clock drivers are legalized taking fixed cells, macros, blockages, and other clock cells into account. When after legalization the distance of the cell center to the preferred location is larger than the distance specified with the **-max_displacement** option, no driver will be inserted near that preferred location. By default, no maximum displacement is in effect, and cells will be legalized as close as possible to the preferred location.

-locations ***coordinate_list***

Specifies a list of preferred locations where cells should be placed. The cells are legalized such that the center of the inserted cells is as close as possible to these coordinates.

-boxes ***integer_pair***

The **-boxes** option accepts a list of two positive integers defining the number of columns and rows of a location grid. The bounding box of the boundary is subdivided in equally sized boxes according to the specified number of columns and rows. Then the preferred locations are formed by the centers of these boxes. The boundary can be specified explicitly using the **-boundary** option. If not specified, the boundary of the core area is used.

-boundary ***poly_rect***

Specifies the area in which the inserted cells should be placed. If the preferred location of any of the cells to be inserted is not covered by the area enclosed by the boundary, then the cell will not be inserted. The default boundary is the boundary of the core area.

-lib_cells ***lib_cell_list***

Specifies a list of buffer lib_cells that can be used as clock driver cells. All specified lib_cells should have lib_cell purpose **cts** included. For each preferred location, the corresponding voltage area is queried, and the **first** lib_cell in the list that qualifies for that voltage area will be used for the location. If none of the provided driver types qualifies, then no buffer is inserted at that location. This options is mutually exclusive with the **-template** option if the lib cell of the template is neither a buffer nor an inverter.

-template

Specifies that the existing driver cell of the load net (the template cell) should be cloned to create clock driver cells. Using a template cell, it is possible to have inverters or clock gates as clock drivers. All newly created cells will be inserted in same hierarchy level as the original template cell, and are only inserted if the preferred location is in the same voltage area as the original driver. This option is mutually exclusive with the **-lib_cells** option if the lib cell of the template is neither a buffer nor an inverter.

-input_pin

Only valid in combination with **-template**. It specifies a list of pairs of hierarchy and the clock input pin of the template cell. This option is required if the template cell has multiple inputs (Example: clock gating cell, multiplexer etc.) and the template is for a

level higher than 1 (see **-configuration**). Otherwise it is optional. In the Multi-Level Physical Hierarchy flow, the input pin must be specified for each physical hierarchy where a multi-input template cell exists. A list of pairs of hierarchy name and respective clock pin name needs to be specified. For top level the keyword "TOP" should be used as the hierarchy name.

-hierarchy *hier_list_per_block*

Specifies a list of pairs of physical hierarchy and corresponding logical hierarchy in which the clock drivers should be inserted. This option is invalid in conjunction with the **-template** option. For top level the keyword "TOP" should be used as hierarchy name. The logical hierarchy may be specified for few physical blocks and will be auto determined for physical blocks for which the information is not specified.

-short_outputs

Specifies that the output pins of the drivers should all be connected to the same physical net. In this way, multi-driven clock meshes or clock straps can be supported.

-output_net_name *string*

Specifies the name of the shorted output net. This option is only valid when option **-short_outputs** is given.

-transfer_wires_from *net*

Specifies an existing net with shapes and/or vias. These shapes and vias will be moved from the existing net to the shorted output net of the drivers. This option is only valid when option **-short_outputs** is given.

-configuration *config_list*

Specifies multiple levels of clock drivers to be inserted. Each entry in the *config_list* corresponds to a clock driver level. An entry is itself a list of options. The following options can be used in a configuration entry:

-level *integer*
[-prefix *string*]
[-keepouts *poly_rect_list*]
[-max_displacement *distance*]
[-locations *coordinate_list*]
[-boxes *integer_pair*]
[-boundary *poly_rect*]
[-lib_cells *lib_cell_list*]
[-template]
[-input_pin *hier_in_pin_list*]
[-hierarchy *hier_list_per_block*]
[-short_outputs]
[-output_net_name *string*]
[-transfer_wires_from *net*]

The required **-level** option accepts a positive integer indicating the level for the configuration entry. Level 1 is the level near the root, and the level number counts up going towards the leaves of the multi-level clock driver structure. The semantics of the other options is the same as for the corresponding option at the command level. For **-prefix**, **-keepouts**, **-max_displacement**, **-boundary**, and **-lib_cells**, the default value is the value of the command-level corresponding option.

DESCRIPTION

The **create_clock_drivers** command inserts a number of clock buffers into the netlist using the reference library cell specified with the **-lib_cells** option. If you specify more than one buffer library cells, for each inserted buffer, the tool inserts the first usable buffer in the list. The library cell selection process takes power domain and driver/load characteristics into account. The **-loads** option takes a single net or a set of load pins. If a net is specified, then all load pins of that (physical) net form the load pins of the buffers. If load

pins are specified explicitly, then all the specified pins should initially be connected to the same (physical) net. However, not all sink pins of a net may be specified as load pins. The unspecified load pins of the net will not be driven by the inserted buffers, but will remain connected to the original net. All specified load pins will be connected to a single inserted clock driver cell. A subsequent tap assignment step is required to distribute the loads over the inserted buffers. The inputs of the inserted repeaters will connect to the original (physical) net of the load pins. If option **-hierarchy** is given, then all buffers will be inserted in the specified hierarchical cell. Otherwise, the tool will choose an appropriate hierarchy level to insert the cells. The number of inserted repeaters is implicitly defined by specifying a number of ideal (preferred) locations. Repeaters will be inserted at legal locations near the specified ideal locations. During legalization, only fixed cells, marked clock cells, and blockages are considered when determining the legal locations. So, inserted repeaters may overlap with existing data path cells or flipflops. In order to end with a legalized design, run the **legalize_placement** command after the **create_clock_drivers** command. Locations can be specified explicitly by using the **-locations** option and/or implicitly through a regular pattern using **-boxes** and **-boundary**. The default boundary is the core area of the block. The **-boxes** option accepts two positive integers defining the number of columns and rows of a location grid. The bounding box of the boundary is subdivided in equally sized boxes according to the specified number of columns and rows. Then the ideal locations are formed by the centers of these boxes. The ideal locations are pre-filtered using the keepout regions and the boundary. If an ideal location is covered by a keepout region or not covered by the boundary area, then that location is not used. The ideal locations define preferred location of the centers of the cells. So, legalization will try to get the center of the buffers as close as possible to the specified locations. If **-max_displacement** is defined and a legalized cell would land more than the specified distance from the ideal location, then the buffer cell will not be inserted. The inserted buffers will have the **physical_status** attribute set to **fixed** and the **user_dont_touch** attribute set to **true**.

Using the **-lib_cells** option, only simple repeaters (buffers or inverters) can be specified. If non-repeater cells (e.g. clock gates) are to be placed in a regular pattern, then a template version of the cells should be included in the incoming netlist. By specifying the **-template** option instead of **-lib_cells** (these options are mutually exclusive if the library cell of the template is neither a buffer nor an inverter), the command is instructed to make clones of the unique driver cell of the load pins. The clone insertion follows the same rules as the buffer insertion with respect to keepouts, legalization, and load assignment. A clone will only be inserted when the ideal location is in the same voltage area as the voltage area of the template cell if the template cell is neither a buffer nor an inverter. The clones will be in the same hierarchical level as the template cell. If the template cell is a buffer or inverter and the location is in a different voltage area and hence power domain this implies that the **opt.common.allow_physical_feedthrough** application option must be set to **true**. The **-hierarchy** option is not supported when **-template** is given. When inverters are used as lib_cells, the polarity of the clock should not change. This implies that in the single-level insertion, no inverters can be specified through **-lib_cells** because that would invert the clock. Instead, an inverter should be added to the incoming netlist and **-template** can be used to clone this existing inverter.

The command can also insert multiple levels of clock drivers using the **-configuration** option. For the lowest level, all loads in a voltage area will be connected to a single clock driver in that voltage area. The other clock drivers in the voltage area will have dangling outputs. A subsequent tap assignment step is required to distribute the loads over the clock drivers. For the higher levels, an assignment of the clock drivers at level N to the clock drivers of level N-1 is performed based on proximity. At these non-leaf levels, drivers that have no loads assigned will not be inserted. So at non-leaf levels, there will be no drivers with dangling outputs. This results in a multi-level structure where only one of the leaf drivers (per voltage area) drives the original loads, and the other leaf drivers have dangling outputs.

In all situations, the polarity of the clock should be preserved at all template levels and at the leaves. This implies that an even number of inverter levels should be inserted in between template levels, as well as an overall even number of inverter levels. It is an error if a polarity change would occur, and the command will not insert any cell in that case.

In a multi-level structure, the building of a level should be aware of what input pins of the downstream level are to be driven. For single-input downstream cells (e.g. buffers or inverters) this is clear. For multi-input cells such as clock gates or multiplexers, the input pin of the template cell should be specified explicitly using the **-input_pin** option. This option accepts a string pair consisting of hierarchy of the template cell and the pin name of the lib_cell to be driven by the upsteam level. Since root level 1 does not have a driving level, it is not required to specify the input_pin name for level 1, even if this is a multi-input template level.

This command also supports Multi-Level Physical Hierarchy (MLPH) flow. Using this flow, the cells can be inserted inside physical blocks present at any level of physical hierarchy. To enable this feature, the application option **cts.multisource.enable_mlph_flow** must be set to true. The locations of the cells can be specified using the **-locations** option or the **-boxes** option. The **-locations** option would take the coordinates with respect to top level of design. Based on the top-level location, the cell would be inserted inside the physical block present at that location. The **-hierarchy** option can be used to specify the logical hierarchy in each physical block which should be used for buffer insertion. For top level, the keyword "TOP" must be used for physical hierarchy block. The **-**

hierarchy option can be specified in global scope as well as per level. The value for a particular level of buffer insertion will take precedence over global scope.

The MLPH flow supports single level cell insertion as well as multi-level cell insertion using the **-configuration** option. Loads of the original clock net are connected to appropriate cells in the single level cell insertion, while they are connected to appropriate cells in the leaf level in the multi-level cell insertion. The load connectivity is done based on the physical hierarchy of the inserted cells as well as the loads. Loads of the original clock net may reside in different physical hierarchies. In this flow, the loads present in a particular physical hierarchy are connected to any one of the cells inserted in that physical hierarchy. If there are no loads in a physical hierarchy, the output pin of all the cells inserted in that block are left dangling. It is recommended, that the cells should be inserted in all physical blocks in which any loads are present. However, in case there are loads present in a physical block but no cell could be inserted in it, then the loads would be connected to a suitable cell present in a different physical hierarchy.

In the multi-level cell insertion, the load assignment of the non-leaf level cells is done based on the proximity of the loads. This may result in a cell present in a physical block driving loads present in other physical blocks. New logical ports may be created in this process if necessary.

The MLPH flow also supports the **-template** option. If the template cells are to be cloned in different physical hierarchies, then the template cell should pre-exist in all those hierarchies. The output net of the template cells in all physical hierarchies should be provided using the **-loads** option. The template cells present in different physical hierarchies may have different masters. The template cell master in a particular physical block is used for cloning cells in that particular physical block. In case a template cell is present in a physical hierarchy but no location falls on top of that physical hierarchy, then the pre-existing template cell would be deleted. The logical hierarchy of the pre-existing template cells is used for cloned cells in the corresponding physical block. For a multi-input template cell, the input pin which should be used for connection in the clock network should be specified for each hierachy using the -input_pin option.

In the MLPH flow, the physical blocks where cells are to be inserted should be editable and port-punching should also be enabled. The optimization of physical blocks must be enabled using the application option **top_level.optimize_subblocks**. In addition, the MSCTS full MV flow should also be turned ON using the application option **cts.multisource.enable_full_mv_support**. An error message would be generated in case the physical block is read-only.

In a typical flow **create_clock_drivers** is used to insert Htree cells/loads. The command **synthesize_multisource_global_clock_trees** is then used to build and route the global clock tree (Htree). This command uses the highest two metal layers allowed for the nets connecting the created cells to route the long distances. These layers can be controlled by using the **set_clock_routing_rules** and **set_routing_rule** commands in conjecture with an explicit nondefault routing rule created with the **create_routing_rule** command. The layers should be within global minimum and maximum layers specified with the **set_ignored_layers** command.

The **create_clock_drivers** command uses the **cts.multisource.enable_pin_accessibility_for_global_clock_trees** application option to control the placement of inserted cells such that they are accessible from the selected Htree routing layers. This ensures good global clock tree (Htree) routing topology and hence the QoR. In cases where command is not used to insert Htree cells/loads, large displacements are observed or cells are dropped it might be useful to set this application option to **false**.

If a clock mesh has been created prior to inserting clock drivers using this command, the tap drives as well as mesh drivers can be snapped close to the clock mesh route shapes. This feature can be enabled using the **cts.multisource.enable_clock_driver_snapping** application option.

To undo the changes made by the **create_clock_drivers** command, use the **remove_clock_drivers** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example creates a grid of 8 by 8 clock driver cells that drive a multi-driven mesh net named clk1_mesh. The drivers

are placed in a regular pattern that covers the full core area (no -boundary option). No drivers will be placed in the bottom-left keepout area. Any existing shapes and vias on net clk1 will be transferred to the newly created mesh net clk1_mesh.

```
prompt> create_clock_drivers -loads [get_net clk1] \
    -lib_cells [get_lib_cells */CKBUF*] \
    -boxes {8 8} \
    -short_outputs \
    -transfer_wires_from [get_net clk1] \
    -output_net_name clk1_mesh \
    -keepouts [list {{0 0} {850 630}} ]
```

The following example creates a grid of 4 by 4 clones of the original driver cell of the load net. On top of that there is an additional clone at the bottom left. If the cells need to move more than 4 um from their preferred location, they will not be inserted.

```
prompt> create_clock_drivers -loads [get_net clk] \
    -template \
    -boxes {4 4} \
    -locations [list {0 0}]\
    -max_displacement 4
```

The following example inserts 4 levels of clock drivers. The first level is a single inverter in the center of the boundary driving 4 inverters at the second level. These drive the CK pin of 16 clones of the original driver of the net. The cloned drivers drive the 64 leaf buffers for which the name starts with HTREE_LEAF. The name of the other inserted driver cells starts with HTREE (the command-level default). For all cells, except the template clones, the maximum allowed displacement is 4 user units. For the template clones, the maximum displacement is 20 user units.

```
prompt> set invs [get_lib_cells */CKINV*]
prompt> set bufs [get_lib_cells */CKBUF*]
prompt> create_clock_drivers -loads [get_net clk] \
    -prefix HTREE \
    -max_displacement 4 \
    -boundary $boundary \
    -lib_cells $invs \
    -configuration [list \
      [list -level 1 -boxes {1 1}] \
      [list -level 2 -boxes {2 2}] \
      [list -level 3 -boxes {4 4} -template -input_pin {{TOP CK}} \
              -max_displacement 20] \
      [list -level 4 -boxes {8 8} -lib_cells $bufs \
              -prefix HTREE_LEAF] \
    ]
```

The following example inserts a single level of clock drivers in 2x2 configuration in the Multi-Level Physical Hierarchy (MLPH) flow.

```
prompt> set_editability -blocks  [get_designs -hierarchical] -value  true
prompt> set_app_options -name top_level.optimize_subblocks -value all
prompt> set_app_options -name opt.common.allow_physical_feedthrough -value true
prompt> set_app_options -name cts.multisource.enable_full_mv_support -value true
prompt> set_app_options -name cts.multisource.enable_mlph_flow -value true
prompt> set bufs [get_lib_cells */CKBUF*]
prompt> create_clock_drivers -loads [get_nets clk] -lib_cells $bufs -prefix TAP -boxes {2 2}
```

The following example inserts a 4 level buffer tree in the MLPH flow.

```
prompt> set_editability -blocks  [get_designs -hierarchical] -value  true
prompt> set_app_options -name top_level.optimize_subblocks -value all
prompt> set_app_options -name opt.common.allow_physical_feedthrough -value true
prompt> set_app_options -name cts.multisource.enable_full_mv_support -value true
```

```
prompt> set_app_options -name cts.multisource.enable_mlph_flow -value true
prompt> set bufs [get_lib_cells */CKBUF*]
prompt> create_clock_drivers -loads [get_nets clk] \
                -lib_cells $bufs \
                -configuration [list [list -level 1 -prefix TAP1 -boxes {1 1}] \
                        [list -level 2 -prefix TAP2 -boxes {2 1}] \
                        [list -level 3 -prefix TAP3  -boxes {2 2}] \
                        [list -level 4 -prefix TAP4 -boxes {4 4}] \
                ]
```

The following example inserts a 4 level buffer tree in the MLPH flow in the specified logical hierachies. Assume there are 2 physical hierachies A and C. The buffers are inserted in logical hierachies A1 and C1 inside blocks A and C respectively. Additionally the logical hierarchy I1/I2 inside top block is used for buffer insertion at top level.

```
prompt> set_editability -blocks  [get_designs -hierarchical] -value  true
prompt> set_app_options -name top_level.optimize_subblocks -value all
prompt> set_app_options -name opt.common.allow_physical_feedthrough -value true
prompt> set_app_options -name cts.multisource.enable_full_mv_support -value true
prompt> set_app_options -name cts.multisource.enable_mlph_flow -value true
prompt> set bufs [get_lib_cells */CKBUF*]
prompt> create_clock_drivers -loads [get_nets clk] \
                -lib_cells $bufs \
                -hierarchy {{TOP I1/I2} {A A1} {C C1}} \
                -configuration [list [list -level 1 -prefix TAP1 -boxes {1 1}] \
                        [list -level 2 -prefix TAP2 -boxes {2 1}] \
                        [list -level 3 -prefix TAP3  -boxes {2 2}] \
                        [list -level 4 -prefix TAP4 -boxes {4 4}] \
                ]
```

The following example inserts a 4 level buffer tree in the MLPH flow. The first 3 levels are inserted using the specified lib cells. The 4th level clones the template cell present in different physical hierarchies. Each of the physical hierarchies where cloning is required must already have a template cell instantiated in the netlist.

In this example there is a top block with 2 sub-blocks namely “A” and “C”. The output net name of the template cell in top block is “clk1”, while in blocks “A” and “C” it is “A/clk1” and “C/clk1” respectively. All these 3 nets should be specified in the -loads option.

```
prompt> set_editability -blocks  [get_designs -hierarchical] -value  true
prompt> set_app_options -name top_level.optimize_subblocks -value all
prompt> set_app_options -name opt.common.allow_physical_feedthrough -value true
prompt> set_app_options -name cts.multisource.enable_full_mv_support -value true
prompt> set_app_options -name cts.multisource.enable_mlph_flow -value true
prompt> set bufs [get_lib_cells */CKBUF*]
prompt> create_clock_drivers -loads [get_nets {clk1 A/clk1 C/clk1}] \
                -lib_cells $bufs \
                -configuration [list [list -level 1 -prefix TAP1 -boxes {1 1}] \
                        [list -level 2 -prefix TAP2 -boxes {2 1}] \
                        [list -level 3 -prefix TAP3  -boxes {2 2}] \
                        [list -level 4 -prefix TAP4 -boxes {4 4} -template] \
                ]
```

The following example inserts a 4 level buffer tree in the MLPH flow. The first 3 levels are inserted using the specified lib cells. The 4th level clones the multi-input template cell present in different physical hierarchies.

In this example there is a top block with 2 sub-blocks namely “A” and “C”. The output net name of the template cell in top block is “clk1”, while in blocks “A” and “C” it is “A/clk1” and “C/clk1” respectively. The template cell present in each of the physical hierarchies is a multi-input cell. In top level hierarchy the input pin of template cell to be used for connection is "a0" while in hierarchies "A" and "C", the input pin to be used for connection is "a1". These are specified using the **-input_pin** option. For specifying the top-hierachy the keyword "TOP" is used.

```
prompt> set_editability -blocks  [get_designs -hierarchical] -value  true
prompt> set_app_options -name top_level.optimize_subblocks -value all
prompt> set_app_options -name opt.common.allow_physical_feedthrough -value true
prompt> set_app_options -name cts.multisource.enable_full_mv_support -value true
prompt> set_app_options -name cts.multisource.enable_mlph_flow -value true
prompt> set bufs [get_lib_cells */CKBUF*]
prompt> create_clock_drivers -loads [get_nets {clk1 A/clk1 C/clk1}] \
                        -lib_cells $bufs -configuration [list \
                          [list -level 1 -prefix TAP1 -boxes {1 1}] \
                          [list -level 2 -prefix TAP2 -boxes {2 1}] \
                          [list -level 3 -prefix TAP3  -boxes {2 2}] \
                          [list -level 4 -prefix TAP4 -boxes {4 4} -template -input_pin {{TOP a0} {A a1} {C a1}}] \
                         ]
```

SEE ALSO

create_clock_straps(2)
synthesize_multisource_clock_taps(2)
synthesize_multisource_global_clock_trees(2)
remove_clock_drivers(2)
remove_multisource_global_clock_trees(2)
opt.common.allow_physical_feedthrough(3)
cts.multisource.enable_mlph_flow(3)

create_clock_rp_groups

Creates relative placement groups for leaf level clock tree cells and immediate drivers.

SYNTAX

int **create_clock_rp_groups**
[-min_sinks *sinks*]
[-max_sinks *sinks*]
[-distance *distance*]
[-timing_driven]
[-max_rp_rows *rows* | -auto_shape]
[-cells *cell_list*]

Data Types

sinks	integer
distance	float
rows	integer
cell_list	collection

ARGUMENTS

-min_sinks *sinks*

Specifies the minimum number of sinks that should be driven by a driver cell, to be considered for grouping. If any driver cell is driving less than the given number of sinks then relative placement group will not be created for driver and sinks. The default value is 2.

-max_sinks *sinks*

Specifies the maximum number of sinks that should be driven by a driver cell, to be considered for grouping. If any driver cell is driving more than the given number of sinks then relative placement group will not be created for driver and sinks. The default value is 128.

-distance *distance*

Specifies the maximum allowed distance between the sinks in one relative placement group. It is manhattan distance in microns. If distance between sinks is more than the given distance then more than one relative placement groups will be created. The default value is 100 microns.

-timing_driven

Specifies that the timing critical sinks should not be considered for grouping. By default all the flops having negative slack are not considered.

-max_rp_rows *rows*

This option controls the shape of the relative placement group and specifies the maximum number of rows. If total number of cells in relative placement group (sinks and driver) are more than the given rows then multiple columns will be created. The default value for the rows is 32. If number of sinks are 128 or more then shape of relative placement group will be decided automatically.

-auto_shape

Specifies that the shape of the relative placement group should be decided automatically. The shape is decided based on the spread of the cells of the relative placement group.

-cells *cell_list*

Specifies the list of names or collection of cells to be clustered. If cells are not given then command will consider all cells of the design for grouping.

DESCRIPTION

The **create_clock_rp_groups** command creates relative placement groups for clock tree sinks. Normally one relative placement group is created for one group of sinks being driven by same clock driver and the drive also becomes part of the relative placement group. It can create more than one relative placement group for sinks of same clock driver depending on the options given to the command.

EXAMPLES

The following example uses the **create_clock_rp_groups** command to create relative placement groups with default values.

```
prompt> create_clock_rp_groups
1
```

The following example uses the **create_clock_rp_groups** command to create relative placement groups, where sinks of one group are not far than 50 microns from each other, and all considered clock drivers drive equal or more than 3 sinks.

```
prompt> create_clock_rp_groups -distance 50 -min_sinks 3
1
```

The following example uses the **create_clock_rp_groups** command to create relative placement groups for clock sinks with maximum 16 rows.

```
prompt> create_clock_rp_groups -max_rp_rows 16
1
```

SEE ALSO

add_to_rp_group(2)
create_rp_group(2)
set_rp_group_options(2)
get_rp_groups(2)
remove_from_rp_group(2)
remove_rp_group_options(2)

remove_rp_groups(2)
write_rp_groups(2)

create_clock_skew_group

Creates a new user-defined skew group.

SYNTAX

collection **create_clock_skew_group**
[-name skew_group_name]
[-mode *mode*]
[-clock *clock*]
-objects object_list

Data Types

skew_group_name	string
mode	collection
object_list	collection

ARGUMENTS

-name *skew_group_name*

Specifies the name of the skew group. If not specified the tool generates a unique one on its own.

-mode *mode*

Specifies the mode for which the skew group needs to be created. If not specified the current mode is assumed.

-objects *object_list*

List of pins and/or ports which need need to be grouped.

-clock *clock*

Specifies the clock for which the skew group needs to be created. If not specified the skew group is applied to all the clocks going through the pins.

DESCRIPTION

This command creates a user-defined skew group. Sink pins specified in a skew group will be balanced against each other to meet the specified desired skew and target early delay.

Incremental addition of sink pins in a skew group definition is not supported.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-mode** option.

EXAMPLES

The following example creates a skew group for the current mode named grp1 with reg_g2/D1/CK and reg_g2/D2/CK as sink pins:

```
prompt> create_clock_skew_group -name grp1 \
  -objects {reg_g2/D1/CK reg_g2/D2/CK}
```

SEE ALSO

report_clock_skew_groups(2)
remove_clock_skew_groups(2)

create_clock_straps

Creates straight clock shapes (straps) and vias to implement clock meshes or clock spines. The straps and vias are attached to existing clock nets. A clock mesh is a two-dimensional grid in a horizontal and a vertical layer, where the straps are connected by vias at the intersection points. Clock spine structures can be either one-dimensional or two-dimensional. One dimensional spines, are straps in a single direction. In a two-dimensional spine structure, each *spine* in the spine direction connects to multiple *stripes* in the orthogonal direction. Shapes connected to one spine however, do not connect to shapes of a different spine. This command can generate meshes, one-dimensional spines and two-dimensional spines. It can also create stripes that connect to automatically detected pre-existing spines.

SYNTAX

status **create_clock_straps**
[-nets *net_list*]
[-boundary *poly_rect*]
[-keepouts *poly_rect_list*]
[-layers *layer_list*]
[-widths *distance_list*]
[-grids *iterator_list*]
[-margins *distance_list*]
[-create_ends]
[-types *type_list*]
[-length *distance*]
[-backoff *distance*]
[-detect_length *distance*]
[-spine_direction *direction*]
[-bias]
[-bias_to_nets *net_list*]
[-bias_margins *distance_list*]
[-allow_floating *true*|*false*]
[-allow_splitting *true*|*false*]
[-clear]

Data Types

net_list	list or collection of nets
layer_list	list or collection of one or two layers
distance	distance in user units
distance_list	list of one or two distances
type_list	list of one or two types in {*user_route stripe detect*}
iterator_list	list of one or two grid iterators
poly_rect	a polygon definition (see create_poly_rect for format)
poly_rect_list	list of poly_rect
direction	*horizontal* or *vertical*

ARGUMENTS

-nets *net_list*

For each of the nets, shapes and vias will be created/deleted according to the specification in the other arguments. Typically, a single net will be specified. Multiple nets are only allowed when the *-clear* option is specified or when one of the specified types is *detect*. The specified logical net(s) should exist.

-boundary *poly_rect*

Specifies the boundary for strap creation. All created shapes are confined to this boundary. The default boundary is the core area of the design.

-keepouts *poly_rect_list*

Depending on the value of the *-allow_splitting* option, straps will be chopped or skipped such that they do not overlap any of the specified poly_rect's in the keepout list.

-layers *layer_list*

This command can create net shapes in horizontal, vertical, or both directions. If straps in both directions are required, then specify the horizontal and the vertical layer. If only straps in one direction are required, then specify the layer in that direction.

-widths *distance_list*

Specifies the width of the straps in horizontal and/or vertical direction.

-grids *iterator_list*

Specifies the locations (grid-lines) of the straps. For each required direction one iterator should be provided. If both directions are required, the first iterator corresponds to horizontal straps and the second iterator to vertical straps. Each iterator has the form *start* or *{start end step}*. In its first form, only one location (grid-line) is specified. In its second form, a set of locations (grid-lines) is specified : start + i * step, for each integer i with (i >= 0) and (start + i*step <= end).

-margins *distance_list*

Specifies the maximum distance of a created strap to its specified grid-line. If it is impossible to create the shape within this distance the shape will not be created. The default value is 0, indicating that the straps should be created exactly on the specified grid-lines. If that is not possible, they will be dropped.

-create_ends

Specifies that straps should be created on the edges of the bounding box of the boundary as well. These straps are additional to the straps as specified by the grid iterator(s). The boundary can be specified with the -boundary option. Note that only the edges in the specified direction(s) will be considered.

-types *type_list*

Specifies the type of straps to be created in the corresponding directions. Each type has the value *user_route*, *stripe*, or *detect*. The value *detect* is only valid when also another value is specified. If one of the values is *detect* then we are implicitly in spine creation mode and not in mesh creation mode. Moreover, *detect* indicates that there are already spine shapes in the *detect* direction and that only straps in the other direction should be created. These created straps will connect to the existing spine shapes by means of vias. The created shapes will have their *shape_use* attribute set to their corresponding type-value. Shapes in the *detect* direction are only considered to be part of a spine if their length is at least -detect_length. Also, considered shapes in the *detect* direction are mapped onto the same spine if their distance is smaller than the specified margin in the *detect* direction.

-length *distance*

Specifies implicitly that we are in spine creation mode. The length specifies the maximum length of the stripes that connect to the detected or created spines.

-backoff *distance*

Specifies implicitly that we are in spine creation mode. The backoff specifies the minimum distance between stripes of different spines.

-detect_length *distance*

This option is only valid if one of the specified types is *detect*. Specifies the minimum length of an existing shape in the *detect* layer in order to consider it a spine shape.

-spine_direction *direction*

Specifies the direction of the spines. This option is not allowed when one of the specified types is *detect*. In that case, the *detect* direction is always the spine direction. This option is required in spine mode if none of the types is *detect*.

-bias

Specifies that shielding with existing power or ground nets is preferred.

-bias_to_nets *net_list*

Specifies that shielding with nets from the net_list is preferred. These nets add to the power and ground nets if the -bias option is used.

-bias_margins *distance_list*

Specifies how far the shielding shape can be from the specified grid-line of the clock shape. If a shielding shape is found within the specified distance, and the shielding length is at least half of the clock shape, then the clock shape will be created next to the shielding shape. The default bias margin is

0. For the case of spines, which is when -types detect, -length, -backoff or -spine_direction is specified, the bias_margin can be a single value or a tuple. if the value is a single value, it applies to the first layer mentioned -in the -layers list. If there is a second layer, its bias_margin will be default i.e. 0.

-allow_floating *true*|*false*

If set to *true*, allow straps in clock meshes or two-dimensional spines that are not connected to a shape in the orthogonal direction. If set to *false*, then straps that do not connect to a shape in the orthogonal direction will be skipped. The default value is *false*.

-allow_splitting *true*|*false*

If set to *true*, allow partial or multiple straps at a specified grid line when blockages or keepouts prevent full-length straps. If set to *false*, only allow full-length straps at the grid lines. If a full-length strap cannot be generated due to blockages or keepouts, the strap will be skipped. The default value is *true*.

-clear

Remove the straps of the specified nets that were created in previous calls of create_clock_straps.

DESCRIPTION

This command creates a set of horizontal and/or vertical straight net shapes (straps) suitable to build a clock mesh or a clock spine structure. A clock mesh is a two-dimensional grid of metal straps in two orthogonal layers. The straps in one layer are connected to

the straps in the orthogonal layer by means of vias or via stacks at the intersection points. A clock spine structure can be one-dimensional or two-dimensional. A one-dimensional spine structure is a set of straps (the spines) in a single layer. A two-dimensional spine structure is a set of *spines* where each spine intersects with a set of *stripes* in the orthogonal direction. Each spine is connected to its stripes at the intersection points by means of vias or via stacks. The straps belonging to one spine do not connect to straps of a different spine. So each strap in the stripe direction is connected to exactly one strap in the spine direction.

The options *-layers*, *-widths*, *-margins*, *-grids*, *-types*, and *-bias_margins* accept a list of one or two values. All these options (if specified) should have the same number of values. If the number of values is one, then one-dimensional spines will be created. If the number of values is two, then a mesh or two-dimensional spines will be created. In the latter case, the first value in each list refers to the horizontal direction, and the second value to the vertical direction. If only one value is given in the lists, then the direction of the specified layer defines the direction of the one-dimensional spines. There is one exception to this rule, however. If one of the two *-types* values is *detect*, then only one grid iterator can be specified. This grid then applies to the non-detect direction.

The mesh or stripe structure covers a recti-linear area which can be specified explicitly with the **-boundary** option. The default area is the core area. The area as specified by the *-keepouts* option will not be covered by clock straps. Note that routing blockages and/or existing shapes will also block area and will prevent clock straps from being created at these locations.

The layers of the straps are specified as a list with the **-layers** option. The specified layers should be routing layers. If two layers are specified, these do not have to be adjacent. Connection between orthogonal non-adjacent layers will be made using via stacks. Typically, however, adjacent layers will be used to create clock meshes or two dimensional spines.

The **-widths** option specifies the width of the created straps in the corresponding direction. This is a required option. Typically, clock straps in meshes or spines are wider than the minimum width for the layer. These wider straps have less resistance and result in faster signal propagation in the mesh or spines.

The locations of the clock straps are specified using the *-grids* option. Each item in the list specifies an iterator that defines equidistant positions in the corresponding direction. The format of an iterator is *{start end step}*. E.g. if the corresponding layer is vertical and the iterator is {350 1350 100}, then the iterator defines vertical grid lines at x-coordinates 350, 450, 550, ..., 1350. The iterator values are in user units. These locations are the preferred locations for the clock straps in the corresponding direction.

The command will try to create clock straps at the grid-lines. However, if these grid-lines are off-track, or if part of the grid-lines is blocked, the command can search in the proximity for a better on-track strap location. The search area is defined by the *-margins* option. This option indicates how far the center line of the created strap can be from the specified grid-line.

When the *-create_ends* option is specified, also straps will be created at the edges of the bounding box of the boundary. These straps are additional on top of the straps defined by the *-grids* option. For a rectangular mesh structure, these end straps will create a ring at the edges of the bounding box. For one-dimensional spines, the *-create_ends* option causes two additional spines to be created at the boundary edges in the direction of the spines. For two-dimensional spine structures, the *-create_ends* option only applies to the stripes that are created orthogonal to the spines. So two additional stripes are created at the boundary edges that correspond to the stripe direction.

The *-types* option specifies the *shape_use* attribute of the created straps and vias. Valid shape_use values are *user_route* and *stripe*. If the value is *user_route*, the comb router is not allowed to connect to the shapes. If the value is *stripe*, then the comb router can connect to these shapes. The special type value *detect* can be used to indicate that no shapes should be created in the corresponding direction, but that existing shapes should be identified as spines, and that only stripes should be created. Shapes of both ports and nets can be detected as spine shapes. For existing net shapes to be detected as spines, their *shape_use* attribute should be set to *user_route* or *stripe*. Port (terminal) shapes with *shape_use* attribute *user_route* or *stripe* will also be detected as spine shapes. When no shapes with shape_use value *user_route* or *stripe* exist on neither the net nor the port then all shapes of the port, irrespective of the shape_use value, will be considered as spines. A value of *detect* implies that a two-dimensional spine structure will be created. Note the terminology here. Stripes are the straps that are orthogonal to the spines. Both spines and stripes can have shape_use *stripe*, although it is typical that the spines have type/shape_use *user_route* and the stripes have type/shape_use *stripe*. When spines are automatically detected, these may belong to different nets. All spines of the specified nets in the *-nets* option will be detected. The stripes of a spine will be attached to the spine's net.

Both the *-length* and the *-backoff* options implies creation of a two-dimensional spine structure. The *-length* option specifies the maximum length of the stripes connected to the spines. The *-backoff* options specifies the minimum distance between stripes of different spines. If the maximum length would violate the backoff value, then the length of the stripe is reduced to honor the backoff value. The default length value is infinite. The default backoff value is twice the minimum spacing rule in the stripe layer.

When the *-types* value is *detect*, the command will identify existing spines. Spine shapes should be in the corresponding layer. The shapes should be owned by either the net or port connected to the net where the straps are created for. If a net or port has shapes with shape_use *user_route* or *stripe* then those shapes will be considered as spines. If no such shapes exist, then any port shape is taken into account, irrespective of the shape_use value. In order for a shape to be considered a spine shape, the width should at least be the corresponding *-widths* value. Multiple spine shapes that are within a distance of the corresponding *-margins* value, are considered to be part of a single pre-existing spine. Only resulting spines with a length longer than the *-detect_length* value are considered as spines for stripe creation.

When a two-dimensional spine structure needs to be created, the spine direction should be specified using the *-spine_direction* option. The spine direction is either *horizontal* or *vertical*.

The location of the created clock straps can be biased such that they are shielded by existing power and ground shapes. Using the *-bias* option, the command will identify power and ground shapes that can be used for shielding. If these shapes are with the corresponding *-bias_margins* value, and the shielded length is at least 50% of the strap length, then the clock strap will be created adjacent to the identified power or ground shape. It is also possible to specify other nets than power and ground nets using the *-bias_to_nets* option. If you use both *-bias* and *-bias_to_nets*, then both power and ground shapes and shapes from the specified bias nets can be used for shielding.

If the value of option *-allow_splitting* is set to *false*, then the command will only create full-length single straps per grid-line. E.g. if a grid-line is partially covered by blockages, keepouts and/or other shapes, then no strap will be created for that gridline. If the value of *-allow_splitting* is set to *true*, then partial and/or multiple straps per grid-line are allowed, and shorter straps may be created in between the blockages. If a stripe strap cannot be connected to a spine shape in the orthogonal direction, then it will not be created unless the *-allow_floating* option is set to *true*. The default value for *-allow_splitting* is *true*.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example creates a grid with horizontal straps on M7 and vertical straps on M8. The length of straps is limited by the specified boundary. It includes the top, bottom, left and right straps that form a ring around the mesh. The strap shapes are owned by the net called clk1_mesh. This net should exist. The command inserts horizontal straps near vertical positions 20, 120, 220, ..., 1120. The maximum deviation to these positions is

10. It inserts vertical straps near horizontal positions 60, 210, 360, ...,

960. The width of the straps is 2.4 in both directions. The shape_use attribute of the created shapes will be set to *stripe* (default). A keepout region is created in the botton-left area. Straps that are overlapping with this keepout region will be chopped due to the '-allow_splitting true' option. With '-allow_splitting false', straps that would overlap with the keepout region will be skipped.

```
prompt> create_clock_straps -nets [get_net clk1_mesh] \
   -boundary {{0 0} {1200 980}} \
   -layers {M7 M8} \
   -widths {2.4 2.4} \
   -grids {{20 1200 100} {60 980 150}} \
   -margins {10 10} \
   -create_ends \
   -keepouts [list {{0 0} {120 240}} ]\
   -allow_splitting true
```

The following example creates a one-dimensional spine structure. Spines are in horizontal layer M7 and have a width of 3.6. The spines are near locations 50, 170, 290, ..., 770. If a power or ground net can be found within 20 distance of the specified location, then the strap is shifted such that is shielded by the power or ground shape. The shape_use attribute of the created shapes will be

set to *user_route*.

```
prompt> create_clock_straps -nets [get_net clk] \
   -layers M7 \
   -widths 3.6 \
   -grids {{50 800 120}} \
   -margins 10 \
   -types user_route \
   -bias -bias_margins 20
```

The following example creates a two-dimensional spine structure. Spines are in vertical layer M8 and have a width of 3.6. The stripes are in horizontal layer M7 and have a width of 2.4. Stripes that cannot connect to their spine will still be created and left floating. The shape_use attribute of the spines will be set to *user_route*.

```
prompt> create_clock_straps -nets [get_net clk] \
   -layers {M7 M8} \
   -widths {2.4 3.6} \
   -grids {{20 1200 100} {60 980 150}} \
   -margins {10 10} \
   -types {stripe user_route} \
   -allow_floating true
```

The following example creates a two-dimensional spine structure based on existing spine shapes in the design (type detect). In effect it will only create the stripes. The created stripes are in horizontal layer M7 and have a width of 2.4. Existing spine shapes should be in layer M8. The minimum width for shapes to be considered as a spine shape is 3.6. When spine shapes are closer than 10 to each other, they will be considered part of a single spine. Only spines with a resulting length of at least 200 will be considered. Since the vertical spines are detected, there is only one grid iterator for the horizontal stripes. The maximum length of a stripe is 110, but if that would violate the backoff value of 5, the stripe length will be reduced. The shape_use attribute of the stripes will be set to *user_route*.

```
prompt> create_clock_straps -nets [get_net clk] \
   -types {user_route detect} \
   -layers {M7 M8} \
   -widths {2.4 3.6} \
   -grids {{20 1200 100}} \
   -margins {10 10} \
   -length 110 \
   -backoff 5 \
   -detect_length 200 \
   -types {user_route detect}
```

SEE ALSO

synthesize_multisource_global_clock_trees(2)
create_clock_drivers(2)
analyze_subcircuit(2)
route_clock_straps(2)

create_command_group

Creates a new command group.

SYNTAX

string **create_command_group** [-info *info_text*]
group_name

ARGUMENTS

-info *info_text*

Help string for the group

group_name

Specifies the name of the new group.

DESCRIPTION

The **create_command_group** command is used to create a new command group, which you can use to separate related user-defined procedures into functional units for the online help facility. When a procedure is created, it is placed in the "Procedures" command group. With the **define_proc_attributes** command, you can move the procedure into the group you created.

The *group_name* can contain any characters, including spaces, as long as it is appropriately quoted. If *group_name* already exists, **create_command_group** quietly ignores the command. The result of **create_command_group** is always an empty string.

EXAMPLES

The following example demonstrates the use of the **create_command_group** command:

prompt> **create_command_group {My Procedures} -info "Useful utilities"**

prompt> **proc plus {a b} { return [expr $a + $b] }**

prompt> **define_proc_attributes plus -command_group "My Procedures"**

```
prompt> help
My Procedures:
  plus

  ...
```

SEE ALSO

define_proc_attributes(2)
help(2)
proc(2)

create_context_for_sub_block

Creates context for block instances.

SYNTAX

status **create_context_for_sub_block**
-block_instance *block_instance_name*
-nets *net_name_list*
physical_model_file_name
[-electric_model_file *file_name*]

Data Types

block_instance_name string
net_name_list list
physical_model_file_name string
file_name string

ARGUMENTS

-block_istance

Specifies the name of the block instance to generate block context.

-nets

Specifies a list of the nets to generate block context.

physical_model_file_name

Specifies the name of the output physical block context model file.

-electric_model_file

Specifies the name of the output electric block context model file. This option is optional.

DESCRIPTION

This command creates context for the specified block instance. The command models the full-chip environment for the block instance, including physical and electrical models:

o Physical model, which describes the location of the connections between block and full-chip designs. These tap locations are

derived by tracing the RedHawk parasitics data. The physical model is generated when the **create_context_for_sub_block** command process is complete, and is written to an output PLOC file.

o Electrical model, which is an effective resistance- and capacitance-based model. The tool uses the native resistance engine to calculate effective resistance which is used to calculate equivalent current during peak dynamic voltage drop analysis. By default, the tool models the full-chip context for the specified blocks. The capacitance data are derived by the RedHawk extraction engine.

o Limitation: this feature only support to create context from Redhawk result

EXAMPLE

The following example shows how to create and use the block context during the block-level voltage drop analysis.

```
prompt> open_block top
prompt> ... # IR analysis setup
prompt> analyze_rail ...
prompt> create_context_for_sub_block -block_instance U1 -nets {VDD VSS} U1.ploc
prompt> create_context_for_sub_block -block_instance U1 -nets {VDD VSS} U1.ploc -electric_model_file U1.context
1

prompt > open_block block
prompt > set_app_options -name rail.pad_files -value U1.ploc
prompt > set_app_options -name rail.block_context_model_file -value U1.context
prompt > analyze_rail ...
```

SEE ALSO

analyze_rail(2)
rail.pad_files(3)
rail.block_context_model_file(3)

create_corner

Creates a corner in the current design.

SYNTAX

string **create_corner**
[-copy *corner_name*]
corner_name

Data Types

corner_name string

ARGUMENTS

corner_name

Specifies the name of the new corner.

-copy *corner_name*

Specifies the name of the corner to copy from.

DESCRIPTION

This command creates a corner in the current design. Corners are used to contain a set of constraints for the design. Unique corners are used to specify different combinations of operating conditions and parasitic parameters. Corners are used together with modes, which are used for different operating modes and power schemes. Timing analysis done for various mode and corner combinations as configured by the **set_scenario_status** command.

Operating condition and parasitics parameters are contained in the current corner. When the design is linked, a single corner named "default" is automatically created and assigned as the current corner. If the **create_corner** command is used, the current_corner is set to the newly-created corner. You can set the current corner to a different corner with the **current_corner** command.

To create a collection of corners matching a pattern and optionally matching filter criteria, use the **get_corners** command. To get a collection of all corners in the design, use the **all_corners** command.

To undo the **create_corner** command, or remove a corner, use the **remove_corners** command.

EXAMPLES

The following example creates a corner named "LowVoltage5" and assigns it as the current corner.

```
prompt> create_corner LowVoltage5
```

SEE ALSO

all_corners(2)
current_corner(2)
get_corners(2)
remove_corners(2)
create_corner(2)
create_mode(2)
set_scenario_status(2)

create_custom_shields

Creates custom shielding for the specified nets with Custom Router, if the nets have shield constraints in effect.

SYNTAX

status **create_custom_shields**
[-nets *nets*]
[-keep_session true | false]

Data Types

nets collection

ARGUMENTS

-nets *nets*

Specifies the collection of nets to be shielded. If not specified, the entire design will be shielded. Nets without shielding constraints will be skipped.

-keep_session *true* | *false*

Specifies whether or not the custom router view of the data will be retained after the command completes. Use true only if you expect to perform additional custom route commands immediately after the current command. The default is false.

DESCRIPTION

The **create_custom_shields** command creates custom shielding shapes for a set of specified nets.

create_cut_metals

This command creates cut metals at metal cut ends.

SYNTAX

status **create_cut_metals**

Data Types

DESCRIPTION

This command creates cut metals at metal cut ends before gdsout. Tool creates cut metals based on the dimensions and the rules defined in the technology file.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

prompt> **create_cut_metals**

SEE ALSO

remove_cut_metals(2)

create_default_topology_plans

Create a default topology plan for a bundle.

SYNTAX

collection **create_default_topology_plans**
[-net_estimation_rule *rule_name*]
[-allow_feedthrough none | select_on | select_off | all]
[-allow_feedthrough_type pure | mixed | any]
[-allow_feedthrough_select *object_list*]
[-allow_flyover none | select_on | select_off | all]
[-allow_flyover_select *object_list*]
[-launch_budget *float*]
[-capture_budget *float*]
[-comment *comment*]
[*bundle_list*]

Data Types

bundle_list collection
object_list collection
rule_name string
comment string

ARGUMENTS

bundle_list

Specifies the list of bundles to create default topology plans. If not specified then it means all the bundles within the current design.

-net_estimation_rule *rule_name*

Specifies the name of the net_estimation_rule of the topology_plan.

-allow_feedthrough none | select_on | select_off | all

Specifies the allow_feedthrough setting of the topology_plan.

"none" indicates that no feedthrough is allowed for the plan.

"select_on" indicates that feedthrough is only allowed through the objects specified in the *-allow_feedthrough_select* option.

"select_off" indicates that feedthrough is allowed through all objects except those specified in the *-allow_feedthrough_select* option.

"all" indicates that feedthrough is allowed through all objects.

-allow_feedthrough_type pure | mixed | any

Specifies the allow_feedthrough_type setting of the topology_plan.

-allow_feedthrough_select *object_list*

Specifies the collection of objects for which the *-allow_feedthrough_select* option is applied.

-allow_flyover none | select_on | select_off | all

Specifies the allow_flyover setting of the topology_plan.

"none" indicates that no flyover is allowed for the plan.

"select_on" indicates that flyover is only allowed over the objects specified in the *-allow_flyover_select* option.

"select_off" indicates that flyover is allowed over all objects except those specified in the *-allow_flyover_select* option.

"all" indicates that flyover is allowed over all objects.

-allow_flyover_select *object_list*

Specifies the collection of objects for which the *-allow_flyover_select* option is applied.

-launch_budget *float*

Specifies the launch budget of the topology_plan.

-capture_budget *float*

Specifies the capture budget of the topology_plan.

-comment *comment*

Specifies the comment of the topology_plan. This is a general-purpose string that can be used for documentation.

DESCRIPTION

The **create_default_topology_plans** command creates a default topology plan for each of the bundle specified by *bundle_list*, or all bundles if not specified, if the corresponding bundle does not have an associated topology plan.

For each bundle that the command is to create a topology plan, there will be one topology node created for the driver block, and one topology node created for each of the load block, and one topology edge per each pair of the node of the driver block and the node of the load block.

If the command line options *-net_estimation_rule*, *-allow_feedthrough*, *-allow_feedthrough_type*, *-allow_feedthrough_select*, *-allow_flyover*, *-allow_flyover_select*, *-lauch_budget*, *-capture_budget*, or *-comment* are provided, they will be applied to the created topology plans as well.

Upon success, the command returns the list of topology plans created.

EXAMPLES

The following example creates default topology plans for all bundles of the current design that do not have associated topology plans yet.

```
prompt> create_default_topology_plans
```

SEE ALSO

create_topology_plans(2)

create_dense_tap_cells

Inserts tap cells with a dense pattern within the specified region. For regions with intensified substrate noise, the command also inserts tap walls along the region boundary.

SYNTAX

status **create_dense_tap_cells**
-lib_cell *cell_name*
-distance *tap_distance*
[-bbox *coordinates*]
[-offset *distance*]
[-pattern every_row | every_other_row | stagger]
[-skip_fixed_cells]
[-prefix *cell_prefix*]
[-separator *cell_separator*]
[-at_distance_only]
[-isn_layer *layer_list*]
[-isn_layer_distance { *{{layer1 distance1 layer_offset1} {layer2 distance2 layer_offset2} ...}*]
[-boundary_lib_cell *cell_name*]
[-horizontal_boundary_spacing *spacing*]
[-left_boundary_flipped]
[-right_boundary_flipped]

Data Types

cell_name string
tap_distance float
coordinates rectangle
distance float
cell_prefix string
cell_separator string
layer_list list
spacing float

ARGUMENTS

-lib_cell *cell_name*

Specifies the library cell that is used as the tap cell. You must specify a single library cell. This is a required option.

-distance *tap_distance*

Specifies the distance in microns between two tap cells in a row. This distance is referred to as the tap distance. This is a required option.

The tap distance specified for this command is typically shorter than the tap distance used by the **create_tap_cells** command, which creates a denser tap cell pattern.

-bbox ***coordinates***

Specifies the rectangular region in which to insert the tap cells. A rectangle is specified as a list of four coordinates, {*{llx lly} {urx ury}*}, which represent the lower-left and upper-right corners of the rectangle in microns.

If you do not use this option or the **-isn_layer** and **-isn_layer_distance** options, the command will error out. This option cannot be specified with the **-isn_layer** and **-isn_layer_distance** options.

-offset ***distance***

Specifies the distance in microns that the tap pattern should be shifted to the right. This option does not set the absolute location of the tap cells, only shifts the existing pattern.

If the offset distance is a multiple of the tap distance, the effective offset is 0. For example, if the tap distance is 10 microns and the offset is 32 microns, the effective offset is 2 (32 - 3x10).

The default is 0.

-pattern every_row | every_other_row | stagger

Specifies the tap cell insertion pattern. The supported patterns are

- **every_row** (the default)
  The command adds tap cells to every row using the specified tap distance. The tap distance specified for this pattern should be approximately twice the design rule distance limit.
- **every_other_row**
  The command adds tap cells to every other row for odd rows only. This reduces the number of required tap cells by approximately half as compared to the normal pattern. The tap distance specified for this pattern should be approximately twice the design rule distance limit.
- **stagger**
  The command adds tap cells to every row. The tap cells on even rows are offset with half the tap distance relative to the odd rows, producing a checkerboard-like pattern. This pattern also reduces the number of required tap cells by approximately half as compared to the normal pattern. The tap distance specified for this pattern should be approximately four times the design rule distance limit.

-skip_fixed_cells

Prevents the command from inserting tap cells in locations that are occupied by fixed cells.

When you use this option, the tool treats a fixed cell like a blockage, which breaks the row and might cause a tap cell to be inserted on each side of the fixed cell. To prevent the insertion of tap cells that overlap fixed cells without breaking the row, use the **-preserve_distance_continuity** option with the **-skip_fixed_cells** option.

By default, the command inserts a tap cell in each calculated tap location, even if it is occupied by a fixed cell.

-prefix ***cell_prefix***

Specifies the prefix for the created tap cells.

When you use this option, the command uses the following naming convention for the inserted tap cells:

tapfiller!*cell_prefix*!*lib_cell*!*number*

By default, no prefix is added and the tool uses the following naming convention for the inserted tap cells:

tapfiller!*lib_cell*!*number*

-separator *cell_separator*

Specifies the separator character that is used when composing the instance name of the tap cell.

The default is "!".

-at_distance_only

Creates tap cells only at the specified distance, distance/2, or distance/4 (stagger mode). Note that using this option can cause DRC violations.

-isn_layer *layer_list*

Specifies the special non-manufacturing layers defined in the technology file that is used to identify regions where the silicon substrate is subject to intensified substrate noise (ISN).

When a design, hard macro, or soft macro has a substrate noise source, an ISN layer rectangle is specified within a certain radius of the noise source.

If you do not use this option or the **-bbox** and **-isn_layer_distance** options, the command will error out. This option cannot be specified with the **-bbox** and **-isn_layer_distance** options.

-isn_layer_distance { *{{layer1 distance1 layer_offset1} {layer2 distance2 layer_offset2} ...}*

Specifies multiple ISN layers, their corresponding tap distances and layers' corresponding shifting value to right for intensified substrate noise regions. The dense tap cells will be inserted in the order of the least dense to the most dense regions. If the layers overlap, the most dense tap distance will prevail. The layer names must not duplicate. When this options is used, the -isn_layer and the -distance options are not allowed. For the input value, "layer" and "distance" value are required, "offset" value is optional and the default value is 0.

If you do not use this option or the **-bbox** and **-isn_layer** options, the command will error out. This option cannot be specified with the **-bbox** and **-isn_layer** options.

-boundary_lib_cell *cell_name*

Specifies the library cell used to create the tap walls. You can specify only a single library cell.

When you use this option, the command performs the following tasks:

1. Inserts vertical tap walls along the left and right edges of the placeable area boundary.
2. Inserts horizontal tap walls along the top and bottom edges of the placeable area boundary.
3. Marks the inserted tap cells as FIXED.

-horizontal_boundary_spacing *spacing*

Specifies the spacing between two tap cells in microns for horizontal tap walls. If the spacing value is not multiple of the site width, it will be rounded to the smaller value that is a multiple of the site width. If the value is less than the width of the tap cell or not specified, there will be no spacing between two tap cells.

The default is 0.

-left_boundary_flipped

Flips the cells of tap walls on the left boundary.

-right_boundary_flipped

Flips the cells of tap walls on the right boundary.

DESCRIPTION

The **create_dense_tap_cells** command inserts a dense pattern of tap cells in the specified regions. Before using this command, you must use the **create_tap_cells** command to perform standard tap cell insertion.

The tap walls and dense tap cells are necessary to ensure proper operation of the digital circuit. The tap walls and cell placement observe the prohibited vertical abutment rules.

Within the specified region, the command performs the following tasks:

1. Deletes the existing tap cells.
2. Reinserts the tap cells using the pitch specified with the **-distance** or **-isn_layer_distance** option.
3. Marks the inserted tap cells as FIXED.

You can also perform dense tap cell insertion in intensified substrate noise (ISN) regions by using the **-isn_layer** or **-isn_layer_distance** option. For details about the insertion process for ISN regions, see the description for the **-isn_layer**, **-isn_layer_distance** and **-boundary_lib_cell** options.

If you specify the **-boundary_lib_cell** option, the command also inserts tap walls along each region boundary of specified region. A tap wall is a row or column of identical tap cells placed linearly. A tap cell is a special cell with a well tie, substrate tie, or both. Tap cells are typically used when most or all standard cell in the library contain no substrate or well taps.

EXAMPLES

The following example inserts dense tap cells in the ISN region.

```
prompt> create_dense_tap_cells -lib_cell myLib/Cell1 \
  -isn_layer LUP_075 -distance 30
```

SEE ALSO

create_interior_tap_walls(2)
create_exterior_tap_walls(2)
create_tap_cells(2)
create_tap_meshes(2)

create_density_rule

Creates a density rule in a block or technology file.

SYNTAX

collection **create_density_rule**
[-library *library*]
[-tech *tech*]
[-max_density *intValue*]
[-min_density *intValue*]
[-window_size *floatValue*]
-layer *layer_name*
[-max_gradient_density *intValue*]

Data Types

library collection
tech collection
floatValue float
layer_name string
intValue integer

ARGUMENTS

-library *library*

Specifies the library in which to create the density rule. The density rule is added to the technology data contained by or referenced by the specified library.

If you do not specify this option or the **-tech** option, the density rule is created in the technology data of the current library.

-tech *tech*

Specifies the technology data in which to create the density rule.

If you do not specify this option or the **-library** option, the density rule is created in the technology data of the current library.

-max_density *intValue*

Defines the maximum percentage of metal allowed in the window.

-min_density {*intValue*}

Defines the minimum percentage of metal allowed in the window.

-window_size {*float value*}

Defines the size of the window for the density check.

By default, the window step size is half of the defined window size.

-layer ***layer_name***

Defines the metal layer for which the rule is specified.

This is a required option.

-max_gradient_density {*floatValue***}**

Specifies the maximum allowable change in fill density between adjacent windows.

DESCRIPTION

The **create_density_rule** command creates a new density rule in the specified technology data and returns a collection of the created object.

The **signoff_create_metal_fill** command uses the density rules during metal fill insertion.

Typically, density rules are defined in the **DensityRule** sections in a technology file. The technology file should contain a **DensityRule** section for each metal layer.

EXAMPLES

The following example creates a density rule in the technology data of the current library.

```
prompt> create_density_rule -max_density 61 -min_density 11 \
  -layer M1 -window_size 101.00 -max_gradient_density 21
```

SEE ALSO

get_density_rules(2)
remove_density_rules(2)
signoff_create_metal_fill(2)

create_design_rule

Creates a design rule (design_rule) in the specified technology object.

SYNTAX

collection **create_design_rule**
[-tech *tech_object*]
[-library *library*]
-layer1 *layer*
-layer2 *layer*
-min_spacing *distance*

Data Types

tech_object collection
library collection
layer collection
distance float

ARGUMENTS

-tech *tech_object*

Specifies the technology object in which design rule need to be created.

The **-tech** and **-library** options are mutually exclusive; you can specify only one. If you do not specify either option, the rule is created in the technology object of the current library.

-library *library*

Specifies the library in which to create the design rule. The command creates the rule in the technology object associated with the specified library. In an implementation tool, the library is a design library. In the library manager, the library is a cell library. You can specify the library using the library's name or a collection that contains the library.

The **-tech** and **-library** options are mutually exclusive; you can specify only one. If you do not specify either option, the rule is created in the technology object of the current library.

-layer1 *layer*

Specified the first layer of two layers between which design rule need to be created. This option takes collection as input. User must provide valid layer object or layer number as input.

-layer2 *layer*

Specified the second layer of two layers between which design rule need to be created. This option takes collection as input. User must provide valid layer object or layer number as input.

-min_spacing *distance*

Specified the minimum spacing between the layers on which design rule exists. This option takes distance as input i.e. floating point number that represents the spacing in microns.

DESCRIPTION

This command creates a new design rule in the specified technology object. The tool treats the rule same as a rule defined in the DesignRule section of the technology file. For more information about defining design rules, see the "DesignRule Section" in the *Synopsys Technology File and Routing Rules Reference Manual*.

If the command is successful, it returns a collection that contains the created rule; otherwise, it returns an empty string. If there is a command syntax error, it returns a TCL_ERROR.

EXAMPLES

The following example creates a design rule in the technology object of the current library.

```
prompt> create_design_rule -layer1 [get_layers M3] \
  -layer2 [get_layers M4] - min_spacing 1.1
```

SEE ALSO

get_design_rules(2)
remove_design_rules(2)
report_design_rules(2)
set_design_rule_attribute(2)

create_dff_trace_filters

Adds filter patterns to the list of filters used by the **compute_dff_connections** command

SYNTAX

create_dff_trace_filters
-type pin | net
-patterns *list_of_patterns*
-filename *filename*
[-blocks *list_of_blocks*]

Data Types

list_of_patterns list
filename string
list_of_blocks list

ARGUMENTS

-type pin | net

Specifies the type of filter to add, either *pin* or *net*. This indicates which type of objects the patterns will be matched against: either pins (and ports) or nets.

-patterns *list_of_patterns*

Specifies a list of patterns to add to the indicated filter type list. A pattern can be a local name (path relative to the parent block cell), or a global name for the top block. Patterns support the simple wildcard characters "*" and "?". Regular expressions are not supported. Patterns are NOT verified at insertion time and it is possible to insert patterns that do not match anything in the current design.

-filename *filename*

Specifies that the set of filter patterns is read from a file. The file is generated by a previously saved trace filter file written by the **write_dff_trace_filters** command. This option is mutually exclusive with all other options.

-blocks *list_of_blocks*

Specifies a list of block names to use as the base name for this pattern when searching for pin/port/net objects. If **-blocks** is not specified, the default implies a special "all blocks" list, and the associated patterns will be tried against all blocks in the current design. Block names do not support wildcard characters. Block names are not verified at insertion time. It is possible to insert block names that do not match any blocks in the current design.

DESCRIPTION

This command creates trace filter patterns used by the Data Flow Flylines (DFF) tracer command **compute_dff_connections**. Any patterns inserted into this list are applied by the DFF tracer as "stop pins/ports" or "stop nets". Any pin/port or net matched by a pattern will stop the DFF tracer from proceeding any further down this path segment.

A GUI interface to manage filter patterns is available as part of the DFF Gui component's Reload option.

BLOCK NAMES, PATTERNS and ABSTRACTS/MIBs

Block names should represent the names of physical blocks used in the main design. For example, the output of **get_attribute [get_designs] name** provides valid block names. Patterns should then be relative to the block, rather than to the global instance name (unless they belong to the top-level design). For example, if an abstract design B1 is linked in the current design as TOP/MORE/INST_B1 and there exists a macro pin TOP/MORE/INST_B1/M1/p1, then to add this pin as a filter pin, the arguments would be: **-block B1 -patterns M1/p1**.

This separation of reference block name and instance name is necessary when tracing through abstracts. When abstracts are loaded in design view for tracing, they are loaded as the main design of that session, and so the instance prefix is discarded. That is, when abstract B1 is traced, the correct path to the macro pin is just M1/p1 with no other path prefix. Separating the patterns in this manner makes this transition seamless.

If abstract design B1 is also a multiply instantiated block (MIB), the situation is even more complicated. There could exist two instances of block B1 at TOP/MORE/INST_B1a and TOP/MORE/INST_B1b. If a filter pin pattern was now specified as TOP/MORE/INST_B1a/M1/p, you might expect that only the macro pin in INST_B1a would be filtered, but not the one in INST_B1b. However, block B1 is only traced one time, and it cannot be traced both with and without the filter pin. That is, filtering a pin in only SOME instances of a MIB is not supported to avoid confusion. It would be possible if the block was in design view, but then fail if the block was later switched to an abstract view. To avoid this problem, this type of unpredictable pattern is discouraged. When using the **-filename** option, a warning is issued if a pattern is read-in that matches only SOME instances of a MIB by using a global path.

The special "all blocks" list is available for global patterns like matching all scan pins in the design, if they have a specific character pattern that can match them like "SCAN". One pattern for each level of the netlist is still required. Since this could easily match unwanted pins or nets, its use should be reserved for special situations.

PATTERN MATCHING

When the DFF tracer runs, it will process each pattern in turn, and for each block that has an instance in the current session, it will apply the pattern to the instance path of that block. This means that the same pattern file can be used with a design with physical hierarchy, whether the blocks are in design view or abstract view. Switching back and forth should not affect the interpretation of filter patterns.

GENERAL USAGE

This command prepares a list of filter patterns to be used by the **compute_dff_connections** command when tracing through the netlist. The block names and patterns are not verified to exist at insertion time. Rather, at runtime, only those patterns matching a netlist object will be applied. This allows the use of only global filter patterns file for an entire project, whether working on the whole design, or just one block.

EXAMPLES

The following example adds a pattern "M1/*" to the trace filter list to filter out all pins of macro M1 in block p0.

```
prompt> create_dff_trace_filters -type pin -blocks p0 -patterns "M1/*"
```

SEE ALSO

compute_dff_connections(2)
remove_dff_trace_filters(2)
write_dff_trace_filters(2)

create_dft_netlist

Creates a DFT netlist from the current design.

SYNTAX

status **create_dft_netlist**
-type extest

ARGUMENTS

-type extest

Specifies the type of DFT netlist to create. There is only one valid type supported:

- **-type extest**

  This DFT netlist type removes all logic in the design except that needed to operate in the outward-facing EXTEST mode. Only the following logic types are kept: wrapper chains and wrapper control logic, interface logic between wrapper chains and I/O ports, test mode decode logic, and 1500 controller and any other logic required for EXTEST operation.

  This DFT netlist type is supported only for wrapped cores.

DESCRIPTION

The **create_dft_netlist** command creates a DFT netlist from the current design. Currently only one type, **extest**, is supported

This command modifies the current design in memory. After running it, write out the DFT netlist file using the **write_verilog** command. Because the **create_dft_netlist** command removes logic from the design in memory, these should be the last steps in your core creation script.

This command cannot be used in an ILM generation flow.

EXAMPLES

The following example creates and writes out an EXTEST-only netlist: after writing the full-netlist design files:

```
# insert DFT IP and compile
insert_dft
```

```
compile

# write block-level design
save_block -as post_compile

# write an EXTEST-only netlist
create_dft_netlist -type extest
write_verilog block_1_EXTEST.vg
```

SEE ALSO

save_block(2)
set_wrapper_configuration(2)

create_die

Instantiates a die or a stack of die instances in the current design

SYNTAX

collection **create_die**
-location *coordinate*
-z_offset *value*
-orientation *orientation*
[-stack *number*]
die_name
die_block_reference

Data Types

coordinate	list
value	int
orientation	string
number	int
die_name	string
die_block_reference	string

ARGUMENTS

-location

Specifies the location of the die in the current design. The values of x and y are in the unit of micron. The value of z is a non-negative integer with lower z value at the bottom. For a die stack, specifies the bottom-most z location. All dies in the stack have the same x and y location with one level z increment.

-z_offset *value*

Specifies the z location of the die.

-orientation *orientation*

Specifies the orientation of bond pads in the array. Value can be one of: **R0, R90, R180, R270, MX, MXR90, MY, MYR90, N, W, S, E, FN, FW, FS, FE**

-stack *number*

Specifies the number of dies in the die stack. If unspecified, only one die is created.

die_name

Specifies the die name.

die_block_reference

Specifies the referenced die block.

DESCRIPTION

The **create_die** command instantiates a die or a stack of die instances in the current design which is a 3DIC environment. A stack of dies has identical process technology, width, height, x and y locations. The *die_block_reference* might be a die block that has been previously created i with the *create_die_block* command.

EXAMPLES

The following example creates a new block named top in the current library created by **create_lib**.

```
prompt> create_die si_interposer_inst interposer:si_interposer -location {0 0} -z_offset 0 -orientation N
{si_interposer_inst}
```

SEE ALSO

create_die_block(2)
create_bump_block(2)
create_3d_top_design(2)

create_die_block

Creates a die block in the target library.

SYNTAX

collection **create_die_block**
[-force]
-dimensions *dimensions_list*
[-origin *coordinate*]
[-library *lib_name*]
[-design_type *design_type*]
die_block_name

Data Types

dimensions_list list
coordinate list
lib_name string
design_type string
die_block_name string

ARGUMENTS

-force

Overwrites the block if one with the same name already exists in the current library.

-dimensions *dimensions_list*

Specifies the width and height of the die block in micron.

-origin *coordinate*

Specifies the coordinate of origin.

-library *lib_name*

Specifies the target library that contains the process technology of the die block to be created. Creating the die in the current library is not recommended as the current design is a 3DIC environment that might contain multiple process technology.

-design_type *design_type*

Specifies the type of the die. The value can be one of: **bridge**, **die**, **interposer**, **substrate**. The default is **die** type. An interposer is a large die that encompass all other dies in a 2.5D package. A bridge is a small die that offers lateral interconnect within a small local region between two dies. A substrate is a multiple metal layer non-silicon substrate for fanout RDL routing.

die_block_name

Specifies the die block name.

DESCRIPTION

The **create_die_block** command creates a die block in the target library. The die block created is empty and should be populated later.

EXAMPLES

The following example creates a new block named si_interposer in the library top.exp.

```
prompt> create_lib top.exp
{top.exp}
prompt> create_die_block si_interposer -force -dimensions {15000 10000} -library top.exp -design_type interposer -origin {0
Information: Creating block 'si_interposer.design' in library 'top.exp'. (DES-013)
{top.exp:si_interposer.design}
```

SEE ALSO

create_3d_top_design(2)
create_bump_block(2)
create_die(2)
create_lib(2)

create_differential_group

Creates an analog constraint group with differential_pair or differential_group intent in the current design.

SYNTAX

collection **create_differential_group**
[*constraint_group_name*]
-for *objects*
[-twist_style default | diagonal | orthogonal | none]
[-twist_interval *distance*]
[-twist_offset *distance*]
[-gap *distance*]
[-shield_placement default | double_interleave | half_interleave | interleave | outside]
[-valid_layers *layers*]
[-layer_spacings *layer_spacing_pairs*]
[-layer_widths *layer_width_pairs*]
[-force]

Data Types

constraint_group_name string
objects collection
distance float
layers collection
layer_spacing_pairs list
layer_width_pairs list

ARGUMENTS

constraint_group_name

Specifies the name of the constraint group. By default, the command generates a unique constraint group named "differential_pair_n", where n is a unique, monotonically increasing integer value.

-for *objects*

Specifies the net objects in the analog constraint group. The *objects* argument can be a collection of objects or a string that contains the name of the objects.

-twist_style default | diagonal | orthogonal | none

Specifies how the nets of the constraint group will be twisted. The allowed values are default, diagonal, none and orthogonal. Default is none.

-twist_interval *distance*

Specifies the twist interval between the nets in the constraint group. This is applicable only when twist style is not none. The default is zero.

-twist_offset ***distance***

Specifies the twist offset for the nets in the constraint group. This is applicable only when **-twist_style** is not none. The default is zero.

-gap ***distance***

Specifies the spacing between the outermost bit nets and other shapes in the block. The default is zero.

-shield_placement default | double_interleave | half_interleave | interleave | outside

Specifies how the main bus trunk will be shielded if one or more nets of the constraint group are shielded. Allowed values are default, double_interleave, half_interleave, interleave and outside. The default is "default".

-valid_layers ***layers***

Specifies the list of layer names to be used for the main trunk. If specified, layers type is set to range, otherwise it is set to default. When specified, the min and max layers are derived from the list.

-layer_spacings ***layer_spacing_pairs***

Specifies the layer and spacing pairs used in trunk routing. This is a list of alternating layer and spacing, for example, {M1 3.0 M2 4.0}.

-layer_widths ***layer_width_pairs***

Specifies the layer and width pairs used in trunk routing. This is a list of alternating layer and width, for example, {M1 3.0 M2 4.0}.

-force

Deletes an existing constraint group if the name matches the specified *constraint_group_name*. The default is false and the command exits with an error if the name already exists.

DESCRIPTION

The **create_differential_group** command creates a constraint group for a collection of nets with differential_pair or differential_group intent. You must specify two or three nets with the **-for** option. When two nets are specified, the nets form a differential_pair. When three nets are specified, the nets form a differential_group. The constraint is used by Custom Router.

You can specify which layers are valid for the trunk routing and can also specify layer-specific spacing and width values. If one or more nets are shielded, the shielding placement of main trunk can also be specified.

You can specify how the nets in the differential_pair or differential_group can be twisted with the interval and offset.

EXAMPLES

The following example creates a constraint group with a system-generated name of "differential_pair_0" for nets prn and prp which will be of differential_pair intent. The twist style will be none and shield placement will be default. The 'prn' and 'prp' nets are routed with the Custom Router.

```
prompt> create_differential_group -for {prn prp}
{differential_pair_0}
```

The following example creates a constraint group with a system-generated name of "differential_group_1" for nets pr*. Assuming that there are three nets, the nets form a differential_group. The twist style will be none and shield placement will be interleave. A spacing of 3 will be used for layer M2 and spacing of 5 will be used for layer M3. The 'pr*' nets are routed with Custom Router.

```
prompt> create_differential_group -for pr* -shield_placement interleave \
  -layer_spacings {M2 3 M3 5}
{differential_group_1}
```

SEE ALSO

get_constraint_groups(2)
remove_constraint_groups(2)
report_constraint_groups(2)

create_diodes

Creates diodes to fix user-specified antenna violations.

SYNTAX

status **create_diodes**
-options *list_of_violations*

Data Types

list_of_violations list

ARGUMENTS

-options *list_of_violations*

Specifies the antenna violations to be fixed by inserting diodes. You can specify one or more antenna violations. Each antenna violation is described by a list that contains the following elements:

- *port_name*
  Specifies the name of the port with the antenna violation.
- *block_instance_name*
  Specifies the name of the block instance with the antenna violation. For diode insertion on top level ports, specify name of top design for this entry.
- *diode_frame_view_name*
  Specifies the name of the frame view of the diode to insert.
- *number_of_diodes*
  Specifies the number of diodes to insert.
- *max_routing_layer_name*
  Specifies the name of the maximum routing layer within which the diode should be connected to the antenna violating pin.
- *max_routing_distance*
  Specifies the maximum distance in microns along the route from the diode pin to the antenna violating pin.

Note: To use this command to insert diodes for top-level ports (terminals) and connect to them you must specify top design name for the *block_instance_name* entry.

This is a required argument.

DESCRIPTION

This command inserts diodes to fix user-specified antenna violations. This command does not do any antenna analysis on the nets. It inserts the specified number of diodes and connects them. After this is done, it is assumed that the antenna violations are fixed.

This command supports the use of spare diodes in the design. If a spare diode of the same frame view exists near the specified antenna port within the specified distance, it can be used to make a connection. This feature can be controlled with the **diode_insertion_mode** router detail route option. This command supports reuse of filler cell locations to insert diodes. It can be controlled using the **reuse_filler_locations_for_diodes** router detail route option. This command can be used to create new diodes or reuse diodes that are close to the antenna violating port. If the only legal location or spare diode is far off, the command will not be able to insert or reuse that diode or location. The distance up to which search is made is bounded by local area and is determined by the tool.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This command creates diodes to fix user-specified antenna violations on the A port of the CI cell instance by using two diodes with a FRAM view of Adiode within the metal5 layer and within two microns of the port.

```
prompt> create_diodes -options {{A CI Adiode 2 metal5 2}}
```

This command creates diodes to fix two user-specified antenna violations.

```
prompt> create_diodes -options {{A CI Adiode 2 metal5 2}
{B CI2 Bdiode 1 metal6 3}}
```

This command creates diodes for the ABC top-level port of a design with top cell name TCell.

```
prompt> create_diodes -options {{ABC TCell Adiode 2 metal5 2}}
```

SEE ALSO

set_app_options(2)

create_drc_error

Creates a physical DRC error object.

SYNTAX

collection **create_drc_error**
-error_data *drc_error_data*
-error_type *drc_error_type*
[-status error | fixed | ignored | waived]
[-polygons *shape_list*]
[-polylines *shape_list*]
[-endpoints *coord_list*]
[-points *shape_list*]
[-layers *layers*]
[-objects *objects*]
[-must_fix]
[-direction *direction*]
[-required_spacing *distance*]
[-actual_spacing *distance*]
[-width_required *distance*]
[-height_required *distance*]
[-shape_uses *shape_uses*]
[-pin_edge *pin_edge*]
[-information *information*]

Data Types

drc_error_data collection
drc_error_type collection
shape_list list
coord_list list
layers collection
objects collection
direction string
distance float
shape_uses list
pin_edge integer
information string

ARGUMENTS

-error_data *drc_error_data*

Specifies the error data in which to create the physical DRC error.

-error_type *drc_error_type*

Specifies the error type for which to create the physical DRC error. The error type value becomes available as the **error_type** attribute for the error.

-status error | fixed | ignored | waived

Sets the initial status of the new error. Available values are: *error, fixed, ignored,* and *waived*. If omitted, the status value **error** is used. The status value becomes available as the **status** attribute for the error.

-polygons *shape_list*

Polygon or rectangle shapes which describe the DRC error. The *shape_list* argument is a list of shapes where each shape is a list of x- and y-coordinate pairs: {{*x y*} {*x y*} {*x y*} {*x y*} ... }. The shapes become available as the *polygons* attribute for the error.

-polylines *shape_list*

Describes polylines shapes for DRC error. The *shape_list* argument is list of shapes where each shapes consist of list of x- and y-coordinate pairs: {{*x y*} {*x y*} {*x y*} {*x y*} ... }. The shapes become available as the **polylines** attribute for the error.

-endpoints *coord_list*

Specifies the two endpoints of distance shape of violation such as spacing or open error. The *coord_list* argument is a list of two x- and y-coordinate pairs: { {*x y*} {*x y*} }. If more than two points are given, only the first two are used and the rest are ignored. This option is a required option when create open error. The coordinates become available as the **endpoints** attribute for the error.

-points *shape_list*

Specifies points as shape of violation when necessary. The *shape_list* argument is a list of x- and y-coordinate pairs: { {*x y*} {*x y*} ... }. The coordinates become available as the **points** attribute for the error.

-layers *layers*

Specifies layers associated with the given shapes or coordinates. The layers are associated with the given shapes or coordinates. If more than one shape is required to describe the error and an exact association of shapes to layers is needed, provide one shape to the **create_drc_error** command and use the **create_drc_error_shapes** command to add the additional shapes with the associated layers. The layers become available as the **layers** attribute for the error.

-objects *objects*

Specifies design objects associated with the new error. The objects become available as the **objects** attribute for the error.

-must_fix

Sets the **must fix** flag on the error. The default is false. The must fix value becomes available as the **must_fix** attribute for the error.

-direction *direction*

Sets the direction of a spacing violation. The option argument is ignored if the associated error type is not in the spacing class. The error class for the error type is set when it is created. The direction is used in rendering the spacing violation in the GUI layout view. The direction value becomes available as the **direction** attribute for the error.

-required_spacing *distance*

Sets the required spacing distance which was not met in a spacing violation. The option argument is ignored if the associated error type is not in the spacing class. The error class for the error type is set when it is created. The required distance is used to provide a more detailed rendering of the spacing violation in the GUI layout view. The required spacing value becomes available as the **required_spacing** attribute for the error.

-actual_spacing *distance*

Sets the actual spacing distance which did not meet requirements in a spacing violation. The option argument is ignored if the associated error type is not in the spacing class. The error class for the error type is set when it is created. The actual distance is used to provide a more detailed rendering of the spacing violation in the GUI layout view. The actual spacing value becomes available as the **actual_spacing** attribute for the error.

-width_required *distance*

Sets the required width of the drc_error object, which is the required width of the violating object. The required width has meaning in the context of a violation in the "size" error_class, where the width of the object does not meet the requirement.

-height_required *distance*

Sets the required height of the drc_error object, which is the required height of the violating object. The required height has meaning in the context of a violation in the "size" error_class, where the height of the object does not meet the requirement.

-shape_uses *shape_uses*

Sets the shape uses associated with the error. The *shape_uses* values can be used to filter errors in the GUI error browser where the attribute is called "route types". The *shape_uses* become available as the **shape_uses** attribute for the error.

-pin_edge *pin_edge*

Sets the pin edge associated with the error. Given any shape, rectangular or rectilinear shape, the edge numbers are a list of positive integers that start from 1. The lower left-most vertical edge is the starting edge with side number 1. Going clockwise, the side numbers of the next edges are 2, 3, and so on. The pin edge is reported as a part of the detailed textual report in the GUI error browser. The pin edge value becomes available as the **pin_edge** attribute for the error.

-information *information*

Sets the error instance specific information string attribute. Information for an error instance is generally set by setting the brief_info and verbose_info attributes of the owning error type and formatting error instance specific data into these strings. However, there are times that information specific to an error instance must be set. In this case, use this attribute to set the information string.

DESCRIPTION

The **create_drc_error** command creates a physical DRC error object in the given error data for the given error type. Additional shapes with associated layers can be added to a drc_error object after creation by using the **create_drc_error_shapes** command.

Most of the attributes associated with a drc_error object are read-only and can only be set at the time of creation. Exceptions to this is the error status, which can be modified using the **set_attribute** command. All attribute values set at creation can then be accessed with the **get_attribute** command. In addition, the attribute values are displayed in the GUI error browser and the GUI layout view.

Brief and verbose information strings are accessible for each drc_error object as the **brief_info** and the **verbose_info** attributes, respectively. These strings are also displayed in the GUI error browser. These information strings are not attached to the drc_error object in the database but rather with the drc_error_type associated with the drc_error object. A message construction service customizes the information strings for each drc_error object by substituting attribute values from the drc_error object into the strings. A detailed description of the message construction is provided in the man page for **create_drc_error_type**.

EXAMPLES

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then creates an error type in the spacing class. A DRC error instance is then created for the new type.

```
prompt> set data [open_drc_error_data -file_name my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}

prompt> set type [create_drc_error_type -error_data $data \
   -name "route spacing" -error_class spacing \
   -required_objects {net net} \
   -brief_info "A spacing violation between net %net:0% and net %net:1% was detected." \
   -brief_format message \
   -severity error]
{"route short"}

prompt> create_drc_error -error_data $data \
   -error_type $type -objects [get_nets {"F_12_" "F_11_"}] \
   -polygons {{78.8600 189.4600} {79.8800 189.4600} {79.8800 186.3200} {78.8600 186.3200}} \
   -layers METAL -status error -direction vertical \
   -required_spacing 3.400 -actual_spacing 31.400
{"123"}
```

SEE ALSO

create_drc_error_shapes(2)
create_drc_error_type(2)
open_drc_error_data(2)
remove_drc_errors(2)

create_drc_error_data

Creates an error data file and returns a collection that contains the error data object.

SYNTAX

collection **create_drc_error_data**
[-name *name*]
[-file_name *file_name*]
-checker_name *checker_name*
[-checker_version *version_string*]
[-information *info_string*]

Data Types

name string
file_name string
checker_name string
version_string string
info_string string

ARGUMENTS

-name *name*

Specifies the name of the attached error data file to create. The error data file is attached to the current block.

The **-file** and **-file_name** options are mutually exclusive; you must specify one.

-file_name *file_name*

Specifies the name of the external error data file.

The naming convention for an error data file is to append an underscore, the checker name, and the ".err" suffix to the design name. For example, for a design named "my_design" and a checker named "test", the error data file name is named "my_design_test.err".

The **-file** and **-file_name** options are mutually exclusive; you must specify one.

-checker_name *checker_name*

Specifies the name of the rule checking engine creating the error data file.

For example, for the route check performed by the router, the engine name might be "zroute".

-checker_version *version_string*

Specifies the version string for the checker engine.

-information ***info_string***

Specifies an informative message to associate with the error data.

DESCRIPTION

The **create_drc_error_data** command creates the specified error data file. The command fails if the specified error data file already exists.

If created with the **-name** option, the error data file is attached to the current block and is included in the block operation when the block is saved, copied, moved, and so on. To save an error data file created in this manner, run the **save_block** for the associated block.

If created with the **-file_name** option, the error data file is written to disk and is not attached to the current block. The error data file is not included in the block operation when it is saved, copied, moved, and so on. The created error data file is opened in read/write mode. To save an error data file created in this manner, run the **save_drc_error_data** command.

The command returns a collection that contains the error data object associated with the error data file. If no error data object is created, the command returns an empty string.

The **create_drc_error_data** command increments the open count for the error data object. The **create_drc_error_data** (and **open_drc_error_data**) commands must be balanced by an equal number of **close_drc_error_data** commands to close the error data object.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates an attached error data file named "dppinassgn.err", with a checker name of "dppinassgn", and a short information string.

```
prompt> set data [create_drc_error_data -name dppinassgn.err \
  -checker_name dppinassgn \
  -information "This error data contains pin assignment and \
    placement errors for the my_design design"]
{"dppinassgn.err"}
```

The following example creates an external error data file named "my_design_dppinassgn.err", with a checker name of "dppinassgn", and a short information string.

```
prompt> set data [create_drc_error_data \
  -file_name my_design_dppinassgn.err \
  -checker_name dppinassgn \
  -information "This error data contains pin assignment and \
    placement errors for the my_design design"]
{"my_design_dppinassgn.err"}
```

SEE ALSO

close_drc_error_data(2)
collections(2)
get_drc_error_data(2)
open_drc_error_data(2)
remove_drc_error_data(2)
save_block(2)
save_drc_error_data(2)
write_drc_error_data(2)

create_drc_error_shapes

Creates shapes and layers and adds to a physical DRC error object.

SYNTAX

status **create_drc_error_shapes**
-error_data *error_data*
drc_error
[-polygons *shape_list*]
[-polylines *shape_list*]
[-endpoints *coord_list*]
[-points *shape_list*]
[-layers *layers*]

Data Types

error_data collection
drc_error collection
shape_list list
coord_list list
layers collection

ARGUMENTS

-error_data *error_data*

Specifies the error data in which to find the physical DRC error to modify.

drc_error

Specifies the error to which to add the shapes and layers.

-polygons *shape_list*

Polygon or rectangle shapes which describe the DRC error. The *shape_list* argument is a list of shapes where each shape is a list of x- and y-coordinate pairs: {{*x y*} {*x y*} {*x y*} {*x y*} ... }. The shapes become available as the *polygons* attribute for the error.

-polylines *shape_list*

Describes polylines shapes for DRC error. The *shape_list* argument is list of shapes where each shapes consist of list of x- and y-coordinate pairs: {{*x y*} {*x y*} {*x y*} {*x y*} ... }. The shapes become available as the **polylines** attribute for the error.

-endpoints *coord_list*

Specifies the two endpoints of distance shape of violation such as spacing or open error. The *coord_list* argument is a list of two x- and y-coordinate pairs: { {*x y*} {*x y*} }. If more than two points are given, only the first two are used and the rest are ignored. This

option is a required option when create open error. The coordinates become available as the **endpoints** attribute for the error.

-points ***shape_list***

Specifies points as shape of violation when necessary. The *shape_list* argument is a list of x- and y-coordinate pairs: { {*x y*} {*x y*} ... }. The coordinates become available as the **points** attribute for the error.

-layers ***layers***

Specifies the layers associated with the given shapes or coordinates. The layers become associated with the given shapes.

DESCRIPTION

The **create_drc_error_shapes** command adds additional shapes and layers to an existing physical DRC error object in the given error data. Shapes and layers are added to the DRC error object at the time of creation. If more than one shape is required to describe the error and exact association of shapes to layers is needed, use the **create_drc_error_shapes** command to add the additional shapes with the associated layers. The layers become available as the **layers** attribute for the error. The command requires at least one of shape options be specified.

EXAMPLES

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then creates an error type in the basic class. A DRC error instance is then created for the new type. Additional shapes and layers are then added.

```
prompt> set data [open_drc_error_data -file_name my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}

prompt> set type [create_drc_error_type -error_data $data \
   -name "shapely error" -error_class basic \
   -required_objects {net} \
   -severity error]
{"route short"}

prompt> set error [create_drc_error -error_data $data \
   -error_type $type -objects [get_nets {F_12_}] \
   -polygons {{78.8600 189.4600} {79.8800 189.4600} {79.8800 186.3200} {78.8600 186.3200}} \
   -layers METAL]
{"45"}

prompt> create_drc_error_shapes -error_data $data $error\
   -polygons {{74.8600 189.4600} {75.8800 189.4600} {75.8800 186.3200} {74.8600 186.3200}} \
   -layers METAL5
```

SEE ALSO

create_drc_error(2)
create_drc_error_type(2)

open_drc_error_data(2)

create_drc_error_type

Creates a physical DRC error type object.

SYNTAX

collection **create_drc_error_type**
-error_data *drc_error_data*
-name *type_name*
-error_class base | spacing | short | open_locator
[-brief_info *info_string*]
[-brief_format ascii | message]
[-verbose_info *info_string*]
[-verbose_format ascii | message]
[-severity error | information | recommendation | warning]
[-required_objects {layer net pin port cell pinGuide}]
[-num_detected_errors *num_errors*]

Data Types

drc_error_data collection
type_name string
info_string string
num_errors integer

ARGUMENTS

-error_data *drc_error_data*

Specifies the error data in which to create the physical DRC error type.

-name *type_name*

The name for the error type being created. Names must be unique in the scope of the owning error data.

-error_class base | spacing | short | open_locator

Specifies the kind of errors represented by the new error type. The error class determines how the shapes of the associated DRC error instances are rendered in the GUI layout view. It also determines some expected attributes for its error instances. Available values are: base, spacing, short, and open_locator.

The **base** class allows its types to define required associated design objects. The **spacing** class allows its types to define required associated design objects and the error instances to describe a mismatch in actual and required spacing distance. Spacing errors can also be defined with a direction. The **short** class allows its types to define required associated design objects and is rendered with a diagonal cross inside of its shape. The **open_locator** class is for open nets and expects to be associated with a single net and to have two endpoints describing an open in a net as its shape description.

-brief_info *info_string*

Specifies a brief information string for display. Error instances of an error type are shown with brief and verbose information strings that are derived from the brief and verbose information strings associated with its error type. The strings can be customized for the error instance by using tokens that are replaced by the error instance's attribute values. See the DESCRIPTION section for a detailed explanation.

The brief information string is expected to be a compact description that is suitable for display in a tabular format in the GUI error browser.

-brief_format ascii | message

Specifies the display format of the brief information string specified with **-brief_info**. If the format is *ascii*, the brief info string is displayed exactly as specified with the **-brief_info** option, without modification. If the format is *message*, the brief info string is parsed for special tokens. The tokens are replaced with error instance attribute values before display.

-verbose_info *info_string*

Error instances of an error type are shown with brief and verbose information strings that are derived from the brief and verbose information strings associated with its error type. The strings can be customized for the error instance by using tokens that are replaced by the error instance's attribute values. See the DESCRIPTION section for a detailed explanation.

The verbose information string should be a detailed description that is suitable for display in a report format in the GUI error browser.

-verbose_format ascii | message

Specifies the display format of the verbose information string specified with **-verbose_info**. The format can be one of: **ascii** or **message**. If the format is **ascii**, the verbose info string is displayed exactly as specified with the **-verbose_info** option, without modification. If the format is **message**, then the verbose info string is parsed for special tokens. The tokens are replaced with error instance attribute value before display.

-severity error | information | recommendation | warning

Specifies the severity of the message associated with instances of this type.

-required_objects {layer net pin port cell pinGuide}

Specifies the required object classes associated with this error. A type might require that a specific number of design objects be associated with its error instances. For example, a route spacing type error might require that all of its instances have two associated nets. If the spacing violation is between shapes of the same net, the same net can be repeated twice. By specifying required objects, the type allows instances to be filtered for null object association.

The required objects are specified with a list of the design object class names. If multiple instances of a design object type are required, the class name can be repeated. For example, to specify that two nets are required, use the option argument **{net net}**.

While this option allows definition of the required design object types, additional objects can be associated with an error instance in addition to the required design object types.

-num_detected_errors *num_errors*

Specifies the maximum number of detected errors that are allowed. Some checkers allow you to specify a limit on the number of errors detected per type. The checker can add information about how many errors were detected, vs. how many errors were added into the error database, by using this option.

DESCRIPTION

The **create_drc_error_type** command creates physical DRC error objects in the specified error data. Error types must have unique names in the scope of the owning error data and error types must be defined with an error class.

The command supports brief and verbose information strings for error type objects. These strings are displayed for each error instance in the GUI error browser. The strings can be customized with attribute values of an error instance by using the **message** format along with replacement of special tokens with attribute values.

A message construction service parses the error message for delimited tokens. If found, the command substitutes them with attribute values to construct a message appropriate for a specific DRC error instance. The token delimiter is the percent (**%**) character. To pass a percent symbol directly to the output message without modification, escape the character with a backslash: **\%**. A token has the form **%<error instance attribute name[:<ordinal value>]>[.<attribute name>]%** Error instance attributes are named as described below.

For associated design objects, the attribute name is the object type name. If there are multiple objects of the same object type, the specific associated object is identified with the ":<ordinal value>" qualifier. The value starts at zero for the first identified object. For example "pin:0" identifies the first associated pin and "pin:2" identifies the third pin associated with the error instance. If the ordinal value qualifier is omitted, the first object of that type is default. The special character "*" can be used in place of the ordinal value to mean all objects of the given type. The supported object type names are: **layer, net, pin, port, cell,** and **pinGuide**.

In addition, the attribute name **type** can be used to denote the associated error type. Append a period character (**.**) delimiter to the optional attribute name to access a specific attribute for the identified object. If the attribute qualifier is omitted, the object's name is used. Attribute names supported with the associated object are: name and full_name. In addition, pin and port objects support the attribute name **cell** which will print the name of the cell associated with the pin or port. Attribute names supported with the associated error type are: name, severity, and class.

For other error instance attribute fields, access an attribute value by using the attribute name for the object. For example, use **actual** for the actual distance in a spacing violation and **direction** for the direction of a spacing violation. Some of these attributes can also be given with the **:<ordinal value>** qualifier. For example, there can be multiple route types associated with an error. As with design objects, if the ordinal value qualifier is omitted, the first one is selected by default. The * glob character can be used to specify all values. The currently supported error instance attributes are: **status, shapes, bbox, direction, actual, required, routeType,** and **endpoint**.

EXAMPLES

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then creates an error type in the short class.

```
prompt> set data [open_drc_error_data -file_name my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
prompt> create_drc_error_type -error_data $data \
   -name "route short" -error_class short \
   -required_objects {net net} \
   -brief_info "A short between net %net:0% and net %net:1% was detected." \
   -brief_format message \
   -severity error
{"route short"}
```

SEE ALSO

create_drc_error(2)
open_drc_error_data(2)

remove_drc_error_types(2)

create_eco_bus_buffer_pattern

Creates an ECO bus buffer pattern.

SYNTAX

collection **create_eco_bus_buffer_pattern**
-name *name*
-first_buffer left | right | top | bottom
-measure_from left | right | top | bottom
[-distance *length*]
[-repeat_after *number*]
[-user_specified_distance *length_list*]

Data Types

name string
length float
number integer
length_list list

ARGUMENTS

-name *name*

Specifies the name of the ECO bus buffer pattern to create. The name must be unique in a design.

-first_buffer left | right | top | bottom

Specifies the order in which the buffers are created. The order is relative to the location of the first buffer in the pattern. Buffer patterns for horizontal routes are specified by using **-first_buffer top** or **-first_buffer bottom**, and buffers are inserted on the route of the topmost or bottommost bit in the layout. Buffer patterns for vertical routes are specified by using **-first_buffer left** or **-first_buffer right**, and buffers are inserted on the route of the leftmost or rightmost bit in the layout.

-measure_from left | right | top | bottom

Specifies the side of the buffer to measure from when placing the next buffer in the pattern. Buffer patterns for horizontal routes are specified by using **-measure_from left** or **-measure_from right**. Buffer patterns for horizontal routes are specified by using **-measure_from top** or **-measure_from bottom**. The first buffer is placed at the reference side of the pattern, succeeding buffers are placed at the specified distance from the previous buffer in the opposite direction. For example, if you specify **-measure_from left** for a horizontal bus, the second buffer is placed to the right of the first buffer, the third buffer is placed to the right of the second buffer, and so on.

-distance *length*

Specifies the distance between the reference sides of two adjacent buffers. This option is mutually exclusive with the **-**

user_specified_distance option and is used together with the *-repeat_after* option.

-repeat_after ***number***

Restarts the pattern from the reference side after the specified *number* of buffers. If this option is not specified, the pattern continues and does not repeat. This option is mutually exclusive with the **-user_specified_distance** option and is used together with the *-distance* option.

-user_specified_distance ***length_list***

Specifies the distances between successive buffers. The first distance in the list is the distance between the reference sides of the first buffer and the second buffer. The second distance in the list is the distance between the reference sides of the second buffer and the third buffer, and so on. The pattern repeats after the last distance in the list is used, and the next buffer is placed at the reference side of the pattern again.

This option is mutually exclusive with the **-distance** and **-repeat_after** option.

DESCRIPTION

This command creates a buffer pattern with the specified placement edge, buffer-to-buffer placement distances, and pattern size. The command returns an ECO bus buffer pattern, which can be used with the **add_buffer_on_route -user_specified_bus_buffers** command.

EXAMPLES

The following example creates an ECO bus buffer pattern named bpLB. The first buffer is placed at the top left corner. Successive buffers are placed at a 2 micron pitch as measured from the left edge of the buffer. The pattern repeats from the left edge every eight buffers.

```
prompt> create_eco_bus_buffer_pattern -name bpLB \
  -measure_from left -first_buffer top -distance 2 -repeat_after 8
```

The following example creates and ECO bus buffer pattern named bpMF. Distances between buffers are specified with the **-user_specified_distance** option. The pitch between the first buffer and second buffer is 2, the pitch between the second buffer and third buffer is 4, the pitch between the third buffer and fourth buffer is 1. The fifth buffer is placed at the left edge of the pattern.

```
prompt> create_eco_bus_buffer_pattern -name bpMF \
  -measure_from left -first_buffer bottom \
  -user_specified_distance {2 4 1}
```

SEE ALSO

add_buffer_on_route(2)
get_eco_bus_buffer_patterns(2)
remove_eco_bus_buffer_patterns(2)
report_eco_bus_buffer_patterns(2)

create_edit_group

Creates an edit group in the current design.

SYNTAX

collection **create_edit_group**
[-name *edit_group_name*]
[-ungroup_on_remove first | second-to-last | last | never]
[-group_use user | bump_array | macro_array | io_group]
[-cell *cell*]

Data Types

edit_group_name string
cell collection

ARGUMENTS

-name *edit_group_name*

Specifies the name of the edit group.

By default, the command generates a name automatically. The command uses the prefix "EDIT_GROUP_" and appends a number in consecutive order, such as EDIT_GROUP_0, EDIT_GROUP_1, and so on.

-ungroup_on_remove first | second-to-last | last | never

Specifies when the edit group should be automatically removed from the design, depending on when objects in the edit group are removed. The *first* argument specifies that the edit group be removed when any member object is removed. The *second-to-last* or *last* arguments specify that the edit group be removed when the second-to-last or last member objects are removed, respectively. The *never* argument specifies that the edit group remain in the design even after all member objects are removed. By default, the *never* argument is used.

-group_use user | bump_array | macro_array | io_group

Sets the use context of this edit group. The default is *user*.

-cell *cell*

Specifies the physical cell where the edit group is to be added. The edit group is created in the cell's reference block. The cell must reference a block, not a library cell, unless this command is executed in the library manager. In the library manager, edit groups can be created in library cells. When not specified, the edit group is created in the current block.

DESCRIPTION

The **create_edit_group** command groups objects into a named collection. You can use the edit group to define a relative positioning constraint on a collection of objects, so that they are physically transformed in a consistent manner. Edit groups created using this command are containers which are initially empty. Objects can be added to edit groups with the **add_to_edit_group** command, and removed with the **remove_from_edit_group** command. An object can belong to only one edit group at a time. Removing an object from an edit group does not remove the object from the design, however removing an object from the design automatically removes it from its containing edit group. Edit group nesting is permitted, so edit groups can contain other edit groups as member objects.

This command returns the edit group as a single element collection, an empty string if it fails, or a Tcl error if there is a command syntax error.

EXAMPLES

The following example creates an edit group named "edit_group1", which is automatically removed when its last member object is removed.

```
prompt> create_edit_group -name "edit_group1" -ungroup_on_remove last
```

SEE ALSO

add_to_edit_group(2)
get_edit_groups(2)
remove_edit_groups(2)
remove_from_edit_group(2)
report_edit_groups(2)

create_ems_database

Creates an EMS database and returns it in a collection.

SYNTAX

collection **create_ems_database**
ems_database_name

Data Types

ems_database_name string

ARGUMENTS

ems_database_name

Specifies the name for the EMS database to be created. The name should contain the ".ems" suffix.

DESCRIPTION

The **create_ems_database** command creates an EMS database and makes it the current EMS database. The *ems_database_name* argument must honor the naming convention and end with ".ems", such as "cts.ems", "check_design.ems", and so on. Use the **save_ems_database** command after **create_ems_database** to write out the database to disk. The created EMS database name is returned in a collection.

EXAMPLES

The following example creates an EMS database named a.ems and makes it the current EMS database.

```
prompt> create_ems_database a.ems
{a.ems}
prompt> get_current_ems_database
{a.ems}
```

The following example shows the error you receive if you omit the .ems extension.

```
prompt> create_ems_database abc
Error: Illegal EMS filename. Filename needs to have suffix of '.ems' (EMS-004)
```

The following example shows the error you receive if you try to create an EMS database that already exists.

```
prompt> create_ems_database a.ems
Error: file 'a.ems" already exists. It cannot be created. (EMS-032)
```

SEE ALSO

close_ems_databases(2)
get_current_ems_database(2)
get_ems_databases(2)
open_ems_database(2)
save_ems_database(2)
set_current_ems_database(2)

create_ems_message

Creates an EMS message for user-defined EMS rule

SYNTAX

string **create_ems_message**
-rule *rule_name*
[-parameters *parameter_values*]

Data Types

rule_name string
parameter_values string

ARGUMENTS

-rule *rule_name*

Specifies the name of the user-defined rule for which the EMS message needs to be created. This is a mandatory option. If the rule name is invalid or if it does not match any existing user-defined rule name then an error message is displayed.

-parameters *parameter_values*

Specifies the values for the EMS parameters defined in the EMS rule. This is an optional option. The argument is a string, which is a sequence of one or more pairs separated by white-space, where each pair is of the format <parameter_name>: <parameter_value>. All the parameters defined in EMS rule should be provided with a value, or else the creation of EMS message fails and command returns an error message.

DESCRIPTION

The **create_ems_message** command creates an EMS message using the user-defined EMS rule name. Before invoking this command, you should have a current EMS database in memory. If one does not exist then it can be created using the **create_ems_database** command. This command also takes the values for parameters present in user-defined EMS rule.

EXAMPLES

prompt> **create_ems_rule -name "TEMP-001" -severity "Info"** \

```
  -message "Pin:%pin has capacitance:%cap" \
  -parameters "name:pin type:string command:get_pins ; name:cap type:double"
prompt> create_ems_rule -name "TEMP-002" -severity "Error" \
  -message "Cell %cell is not placed" \
  -parameters "name:cell type:string command:get_cells"
prompt> create_ems_rule -name "DUMMY-001" -severity "Warning" \
  -message "This is a dummy warning message without parameters"
prompt> create_ems_database a.ems
{a.ems}
prompt> set cap 0.0004
prompt> create_ems_message -rule "TEMP-001" \
  -parameters "pin:U1/I cap:$cap"
prompt> set cell U23
prompt> create_ems_message -rule "TEMP-002" -parameters "cell:$cell"
prompt> create_ems_message -rule "DUMMY-001"
prompt> report_ems_database
 ...
Information: Pin:U1/I has capacitance 0.0004 (TEMP-001)
Error: Cell U23 is not placed (TEMP-002)
Warning: This is a dummy warning message without parameters (DUMMY-001)
```

SEE ALSO

create_ems_database(2)
create_ems_rule(2)

create_ems_rule

Creates a user-defined EMS rule. An EMS rule is a template used to create EMS messages.

SYNTAX

create_ems_rule
-name *rule_name*
-severity *rule_severity*
-message *message_template*
[-parameters *parameter_properties*]

Data Types

rule_name string
rule_severity string
message_template string
parameter_properties string

ARGUMENTS

-name *rule_name*

Specifies the name used to uniquely identify this rule. This is a mandatory option. If a rule by this name already exists in EMS memory, then an error message will be displayed. The rule name should obey the regular expression [A-Z]+[-][0-9]+ i.e one or more upper-case letters, followed by hyphen character and which should be followed by one or more numerals. If rule name does not obey this regular expression, then an error message is displayed. The sub-string in the rule name before hyphen character is called the **domain** name. If there are many rules which are closely associated with each other then it is advised to choose the same **domain** name.

-severity *rule_severity*

Specifies whether the rule corresponds to an error, warning, or information. This is a required option. The valid values are **error**, **information**, **info**, **warning**, and **warn**. These string values can be in upper or lower case or a mix of both. But it is advisable to stick to one convention for the sake of uniformity. But when the EMS message is displayed in the shell or in the EMS GUI message browser, the severity string displayed will be one of **Error**, **Warning**, or **Information**.

-message *message_template*

Specifies the message template for the rule. This is a mandatory option. The message template can contain placeholders referred to as **parameters**. Parameters begin with a **%** character. The parameter name should obey regular expression [A-Za-z0-9_]+ i.e it should contain one or more of upper or lower case alphabets or numerals or underscore character. There should not be any whitespace between **%** and parameter name. If user wants to use literal **%** the she should use **%%**.

-parameters *parameter_properties*

Specifies the parameter properties for the rule. This is an optional option. If there are no EMS parameters for the rule being defined, then this option is not required. But if there are EMS parameters specified in message template, this option can be used to define the parameter properties. This option takes a string as an argument which contains groups of name-value pairs fields separated by a semi-colon (;). Each group describes properties of an EMS parameter, which again is a sequence of pairs of format <name>:<value>.

Currently, an EMS parameter can have at-most three properties: name-property, type-property and command-property. Here type-property and command-property are optional and name-property is mandatory. name-property will take a valid parameter name as value and which helps EMS to know properties of which parameter are defined. type-property can take one of **string**, **int**, **float** and **double** values. type-property is used to define the type of the EMS parameter. The command-property of the EMS parameter can take value as any of the **get_*** commands such as **get_pins**, **get_cells**, and so on, which take an object name as argument and return the object handle. This property is used to get the object handle corresponding to the EMS parameter name.

DESCRIPTION

The **create_ems_rule** command creates a user-defined EMS rule. An EMS rule is a message template with some auxiliary information used to create the EMS message. The user-defined rules created are not persistent across sessions.

EXAMPLES

```
prompt> create_ems_rule -name "TEMP-001" -severity "Info" \
  -message "Pin:%pin has capacitance:%cap" \
  -parameters "name:pin type:string command:get_pins ; name:cap type:double"
prompt> create_ems_rule -name "TEMP-002" -severity "Error" \
  -message "Cell %cell is not placed" \
  -parameters "name:cell type:string command:get_cells"
prompt> create_ems_rule -name "DUMMY-001" -severity "Warning" \
  -message "This is a dummy warning message without parameters"
prompt> create_ems_database a.ems
{a.ems}
prompt> set cap 0.0004
prompt> create_ems_message -rule "TEMP-001" \
  -parameters "pin:U1/I cap:$cap"
prompt> set cell U23
prompt> create_ems_message -rule "TEMP-002" -parameters "cell:$cell"
prompt> create_ems_message -rule "DUMMY-001"
prompt> report_ems_database
 ...
Information: Pin:U1/I has capacitance 0.0004 (TEMP-001)
Error: Cell U23 is not placed (TEMP-002)
Warning: This is a dummy warning message without parameters (DUMMY-001)
```

SEE ALSO

create_ems_database(2)
create_ems_message(2)

create_exterior_tap_walls

Creates a tap wall at a specified distance outside a boundary edge of a core area, hard macro, soft macro, or hard placement blockage.

SYNTAX

status **create_exterior_tap_walls**
-lib_cell *cell_name*
-bbox *coordinates*
-side top | bottom | left | right
[-x_margin *distance*]
[-y_margin *distance*]
[-prefix *cell_prefix*]
[-orientation *cell_orientation*]

Data Types

cell_name string
coordinates list
distance float
cell_prefix string
cell_orientation string

ARGUMENTS

-lib_cell *cell_name*

Specifies the library reference cell to be used as a tap cell. You must specify a single library cell. This is a required option.

-bbox *coordinates*

Specifies the rectangular tap insertion region in microns. Specify the rectangle as a list of four coordinates, {{llx lly} {urx ury}}, which represents the lower-left and upper-right corners of the rectangle.

A tap wall is created along the longest edge of the core boundary, macro, or hard placement blockage encompassed by the rectangular region. The command issues an error message if it cannot find an edge.

This is a required option.

-side top | **bottom** | **left** | **right**

Specifies the side on which to create a tap wall. You must specify a single side. This is a required option.

-x_margin *distance*

Specifies the margin in microns of a horizontal tap wall.

The default is 0.

-y_margin *distance*

Specifies the margin in microns of a vertical tap wall.

The default is 0.

-prefix *cell_prefix*

Specifies the prefix for the created tap cells.

When you use this option, the command uses the following naming convention for the inserted tap cells:

*<cell_prefix>__<lib_cell>*_R#_C#_*number*

By default, no prefix is added and the tool uses the following naming convention for the inserted tap cells:

extTapWall__<*lib_cell*>_R#_C#_*number*

-orientation *cell_orientation*

Specifies the orientation of the placement of the tap cells. The valid values are: R0, R90, R180, R270, MX, MXR90, MY, MYR90.

The default is R0.

DESCRIPTION

This command adds an exterior tap wall to the design. A tap wall is a row or column of identical tap cells placed linearly. A tap cell is a special cell with a well tie, substrate tie, or both. Tap cells are typically used when most or all standard cells in the library do not contain substrate or well taps.

Exterior tap wall placement does not honor standard cell rows or sites and does not cross the object's boundary. Only one edge of the specified side is be processed in each command execution. If the tap insertion region spans multiple edges, only the longest edge is processed. If the tap insesrtion region does not contain an edge, the command issues an error message.

The tap cells are inserted within the tap insertion region, including the specified margins. A horizontal tap wall starts at the x-margin from the left of the bounding box. A vertical tap wall starts at the y-margin from the bottom of the bounding box. If the tap insertion region, including the margins is too small, the command does not insert tap cells.

You must ensure that the tap insertion region is empty. Otherwise, the inserted tap wall might overlap macros or other cells.

The command marks the inserted tap cells as fixed.

EXAMPLES

The following example creates tap walls outside the longest edge on the bottom of the core area.

```
prompt> create_exterior_tap_walls -lib_cell myLib/Cell1 \
  -side bottom -bbox {{-20 -20} {20 200}}
```

SEE ALSO

create_dense_tap_cells(2)
create_interior_tap_walls(2)
create_tap_cells(2)
create_tap_meshes(2)

create_fill_cell

Creates a new fill cell in the current block.

SYNTAX

status **create_fill_cell**
-reference reference

Data Types

reference string

ARGUMENTS

-reference reference

Name of the internal design in this or another design.

DESCRIPTION

Creates a new fill cell in the current block using the specified reference. The specified reference, must be an internal design. If the specified reference is an internal design from another design, it is copied as an internal design in the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a fill cell, using internal design "m1_fill" from design "myDesign" as reference.

```
prompt> create_fill_cell -create_reference myDesign.m1_fill.design
2
```

SEE ALSO

get_fill_cells(2)
remove_fill_cells(2)

create_frame

Creates a frame view for the current block by extracting blockage, pin, and via information from the design view.

SYNTAX

status **create_frame**
[-block_all auto | true | false | used_layers | *topBlockedLayerName*]
[-block_core_margin *layer_margin_list*]
[-derive_pattern_must_join true | exclude_pg | false]
[-drc_distances *layer_distance_list*]
[-color_based_dpt_flow false | true | auto]
[-connect_within_pin true | false]
[-convert_metal_blockage_to_zero_spacing *layer_distance_list*]
[-create_zero_spacing_blockages_around_pins *layer_width_mode_list*]
[-design_rule_via_blockage_layers *cut_layer_name_list*]
[-enable_via_regions_for_all_design_types true | false]
[-hierarchical true | false]
[-include_nondefault_via *via_list*]
[-include_routing_pg_ports *port_list*]
[-keepout_spacing_for_non_pin_shapes *layer_setting_list*]
[-merge_metal_blockage true | false]
[-pin_channel_distances *layer_distance_list*]
[-pin_must_connect_area_layers *connection_spare_layer_list*]
[-pin_must_connect_area_thresholds *layer_width_list*]
[-port_contact_selections *port_contact_code_list*]
[-preserve_metal_blockage true | false | auto]
[-trim_metal_blockage_around_pin *layer_mode_list*]
[-read_fill_cells true | false]
[-remove_non_pin_shapes *layer_mode_list*]
[-source_drain_annotation *file_name*]

Data Types

topBlockedLayerName string
layer_margin_list list
layer_distance_list list
layer_width_mode_list list
cut_layer_name_list list
via_list list
port_list list
layer_setting_list list
connection_spare_layer_list list
layer_width_list list
port_contact_code_list list
layer_mode_list list
file_name string

ARGUMENTS

-block_all auto | true | false | used_layers | *topBlockedLayerName*

Controls whether the tool creates zero-spacing routing blockages in the frame view for the entire block, except for the area around the pins.

The valid values for this option are

- **false**
  The tool does not create zero-spacing routing blockages.
- **true**
  The tool creates a zero-spacing routing blockage on all layers.
- **used_layers**
  The tool creates a zero-spacing routing blockage only on layers used by the cell.
- **topBlockedLayerName**
  The tool creates a zero-spacing routing blockage only on the specified layer and the layers below it. The layer name must be a metal or poly layer defined in the technology file.
- **auto** (the default)
  The tool determines whether to create zero-spacing routing blockages based on the design type of the cell. For analog, black-box, and module design types, the tool creates zero-spacing routing blockages on all layers (**true**). For macro cells, the tool creates zero-spacing routing blockages only on used layers (**used_layers**). For all other design types, the tool does not create zero-spacing routing blockages.

When the tool does not generate zero-spacing routing blockages in the frame view, it copies all of the physical geometries from the design view to the frame view.

When the tool generates zero-spacing routing blockages in the frame view, it preserves the detailed shapes within the DRC distance from the edges of the routing blockages. The tool generates the metal shapes to cover the area beyond the DRC distance. If a cut shape is covered by generated metal shapes on both the upper and lower metal layers, the tool does not include the cut shape in the frame view. Otherwise, the tool preserves the cut shape. For more information about the DRC distance, see the description of the **-drc_distances** option.

The setting specified by this option applies to all layers. However, if this option setting conflicts with the setting of the **-block_core_margin** option for a layer, the **-block_core_margin** option has priority for that layer.

-block_core_margin *layer_margin_list*

Creates zero-spacing routing blockages on the specified layers with the specified margin between the core blockage and the cell boundary.

Specify this argument using the following syntax:

{ { *layer_name margin* } ... }

The *layer_name* value can be the layer name defined in the technology file or the keyword "used_layers" or "all_layers".

- For "used_layers", the margin value is for the non-empty metal or poly layers.
- For "all_layers", the margin value is for all metal and poly layers.

When there are combined settings, the priority from high to low is:

1) *layer_name*

2) used_layers
3) all_layers

The *margin* value can be zero, a positive number, or a negative number. When the specified margin is 0, the tool blocks the entire cell area, except for the area around the pins, with a zero-spacing routing blockage for the specified layer. A positive margin value specifies the distance between the generated zero-spacing routing blockage and the cell boundary. A negative margin value specifies the distance that the zero-spacing routing blockage extends outside the cell boundary.

The *margin* value can also be a keyword "auto_bloat", which means the tool automatically uses layer negative value of min_spacing of specified layer as the margin value. Therefore, the generated routing blockage is extended outside the cell boundary by min_spacing of the specified layer.

When generating zero-spacing routing blockages in the frame view, the tool preserves the detailed shapes within the DRC distance from the edges of the routing blockages. The tool generates the metal shapes to cover the area beyond the DRC distance. If a cut shape is covered by generated metal shapes on both the upper and lower metal layers, the tool does not include the cut shape in the frame view. Otherwise, the tool preserves the cut shape. For more information about the DRC distance, see the description of the **-drc_distances** option.

If this option setting conflicts with the setting of the **-block_all** option for a layer, this option setting has priority.

-drc_distances ***layer_distance_list***

Specifies the distance for each layer within which to preserve detailed shapes from the edges of a zero-spacing routing blockage generated by the **-block_all** or **-block_core_margin** option.

Specify this argument using the following syntax:

{ { *layer_name distance* } ... }

By default, the tool uses the maximum spacing value from the Layer section of the technology file for each layer.

-color_based_dpt_flow false | **true** | **auto**

Controls whether the tool creates the metal shapes beyond the DRC distance with attribute mask_constraint or not. The metal shapes are added by option block_all or block_core_margin in frame creation.

The valid values for this option are :

- **false**
  The metal shapes beyond the DRC distance are colorless and trimmed by colored shapes.
- **true**
  For metal layers, the metal shapes beyond the DRC distance are colored and trimmed by both colored and colorless shapes.

  For poly and below layers, the metal shapes beyond the DRC distance are still colorless and only trimmed by colored shapes.
- **auto** (the default)
  If the following color rules are defined in the technology file, the option is treated as **true**; otherwise, it is treated as **false**.

  sameColorSpanTblXMinSpacing
  sameColorSpanTblYMinSpacing
  diffColorSpanTblXMinSpacing
  diffColorSpanTblYMinSpacing

-connect_within_pin true | **false**

Controls whether the via enclosure must be within the pin shape.

This option affects via region creation. If **true**, the tool must place the via enclosure inside the pin shape when inserting vias in the

via region. If **false**, the tool can place the via enclosure outside of the pin shape when inserting vias in the via region.

The default is **true**.

-convert_metal_blockage_to_zero_spacing *layer_distance_mode_list*

Converts regular routing blockages on the specified layers in the design view to zero-spacing routing blockages in the frame view, and specifies the distance by which to enlarge the blockage when doing the conversion. Different enlarge distance can be specified for blockages touching pins and not touching pins by setting the mode as 'touch_pin' and 'no_touch_pin' accordingly. Mode 'all' implies the enlarge distance will be applied to all regular routing blockages on the specified layer.

Specify this argument using the following syntax:

{ { *layer_name distance mode*} ... }

The layer name must be a metal or poly layer defined in the technology file. The distance value must be no less than zero. The mode string must be one of 'touch_pin', 'no_touch_pin', or 'all'.

-create_zero_spacing_blockages_around_pins *layer_width_list*

Specifies the width to create zero-spacing routing blockages around pins by layer.

For the pin on cell boundary, the routing blockage will not be generated on the boundary side.

For the shallow pin, the routing blockage will not be generated on the channel area, which is from *pin* to *cell boundary*.

For the deep pin, all sides of the pin will have routing blockages.

Specify this argument using the following syntax:

{ { *layer_name width* } ... }

The width should be no less than zero.

-design_rule_via_blockage_layers *cut_layer_name_list*

Converts zero-spacing routing blockages on the specified cut layers of the frame view to design rule via routing blockages. The option accepts a list of cut layers defined in the technology file and works only for zero-spacing routing blockages on the specified layers.

A design rule via routing blockage is a routing blockage whose **is_design_rule_blockage** attribute is **true**. The purpose of a design rule via routing blockage is to honor the spacing between a via blockage layer and a via layer in the DesignRule section of the technology file. For example,

```
DesignRule {
  layer1 = 'via1Blockage'
  layer2 = 'V1'
  minSpacing = 0.084
}
```

-derive_pattern_must_join true | exclude_pg | false

Sets **pattern_must_join** attribute as **true** to the port which has multiple terminals on one metal layer. Only terminals on metal layers are considered, terminals on cut layers are ignored. The option value controls which type of port should be checked by this option.

The valid values for this option are

- **true**
  The tool derives **pattern_must_join** attribute for the following ports:
- The **port_type** attribute is not power/ground related.

- The **port_type** attribute is power/ground related, and the **is_secondary_pg** attribute is **true**.
- The **port_type** attribute is power/ground related, and the port name is specified in the **-include_routing_pg_ports** option.
- **exclude_pg**
  The tool derives **pattern_must_join** attribute only for the ports which are not power/ground related.
- **false** (the default)
  The tool does not set **pattern_must_join** attribute as **true** to any port.

-enable_via_regions_for_all_design_types true | false

Controls whether to create via regions for all design types.

The default is **false**. Only create via regions for standard cell group design types.

When this option is set to **true**, the tool also checks whether the -block_all or -block_core_margin is set to **true**. And then the tool checks the direction of pin shape direction, and cut channel to the closest boundary on same layer or upper layer base on the direction of pin shape.

- For "vertical" pin:
  If the pin is near to "bottom or top" boundary, create channel on "same" metal layer.
  If the pin is near to "left or right" boundary, create channel on "upper" metal layer.
- For "horizontal" pin:
  If the pin is near to "left or right" boundary, create channel on "same" metal layer.
  If the pin is near to "bottom or top" boundary, create channel on "upper" metal layer.
- For "square" pin:
  Create channel to the nearest boundary on "same" metal layer.

-hierarchical true | false

Controls whether to extract geometries in subblocks.

If **true**, the tool performs hierarchical extraction. If **false**, the tool ignores the geometries in subblocks.

The default is **true**.

-include_nondefault_via *via_list*

Specifies the nondefault vias to consider when calculating the via regions. Specify the nondefault vias by using the contact code names defined in the technology file.

The *via_list* can accept keyword "auto", which means when there is no via region created by default vias, the tool automatically includes the nondefault vias to calculate the via regions.

By default, the extraction process considers only the default vias.

-include_routing_pg_ports *port_list*

Specifies the power and ground port names that require via regions and access edges in the frame view. Usually, these ports are the secondary or bias power and ground ports, which are routed by the signal router.

By default, power and ground ports are ignored when generating via regions and access edges.

-keepout_spacing_for_non_pin_shapes *layer_setting_list*

Creates zero-spacing routing blockages around non-pin shapes on the specified layers with the specified *extension_spacing* from the shape boundary and with the specified *corner_keepout_spacing* from the shape corners.

An *edge_threshold* value is also specified for the layer so that frame can identify "edges" of non-pin shapes. If the created zero-

spacing routing blockages are overlap with pins, the overlap region on the non-edge sides are trimmed by pins.

Specify this argument using the following syntax:

{ { *layer_name* {*edge_threshold extension_spacing corner_keepout_spacing*} } ... }

The *edge_threshold* is for determining edges of shapes. If a segment along the contour of the connected non-pin shape has length smaller than or equal to the threshold, the segment is an edge. But if the segment belongs to concave part of the contour, it is still not an edge. The value should be zero or a positive number. If the value is zero, routing blockages are trimmed by pins on each side of the shape.

The *extension_spacing* is for creating zero-spacing routing blockages around non-pin shapes in direction top, bottom, left and right. The value should be zero or a positive number. If the value is zero, no routing blockage is created.

The *corner_keepout_spacing* is for creating zero-spacing routing blockages around corners of non-pin shapes. The value should be zero or a positive number, and it should be smaller than or equal to the *extension_spacing*. If the value is zero, no corner routing blockage is created.

If **-block_all** is **true** or **-block_core_margin** is specified for the same layer, this option becomes invalid.

-merge_metal_blockage true | **false**

Controls whether to merge metal shapes and regular routing blockages to reduce the number of shapes.

If **true**, the extraction process fills in the space between the two shapes when two nearby geometries are within the spacing threshold, creating a single larger shape for the blockages in the frame view. The spacing threshold is equal to minimum spacing times two plus minimum width.

The default is **false**.

-pin_channel_distances *layer_distance_list*

Specifies the pin channel cutting distance from the block boundary to core area per layer. If the distance of the pin from the block boundary is smaller than the pin channel cutting distance, the tool will cut channel from the pin through shapes or blockages to boundary.

By default, the tool uses layer minWidth + minSpacing from the technology file as the pin channel cutting distance.

-pin_must_connect_area_layers *connection_spare_layer_list*

Specifies connection layer and associated spare layer pairs to designate the required via connection area within the pin geometry.

Specify this argument using the following syntax:

{ { *connection_layer_name spare_layer_name* } ... }

The spare layer can be any layer except a layer with a standard mask name.

By default, the router can connect to any part of a large pin. To restrict the connection to a subarea of a large pin, specify the subarea polygon on a spare layer and specify the spare layer associated with each connection layer using this option.

The following example associates connection layer M1 with spare layer S1 and connection layer M2 and spare layer S2:

{ {M1 S1} {M2 S2} ... }

-pin_must_connect_area_thresholds *layer_width_list*

Specifies connection layer and the width threshold pairs for automatic determination of the required via connection area within the pin geometry.

Specify this argument using the following syntax:

{ { *layer_name width* } ... }

Specify the width threshold using the unit size defined in the technology file.

By default, the router can connect to any part of a large pin. To automatically restrict the connection to a subarea of a pin with various geometry widths, use this option to specify the width threshold for each layer.

For example,

{ {M1 0.56} {M2 0.63} ... }

The tool does not allow via connections in areas within a pin that have a width smaller than the specified threshold width for the layer.

-port_contact_selections *port_contact_code_list*

Specifies the contact codes to use for specific ports during via region generation.

Specify this argument using the following syntax:

{ { *port_name* { *contact_code_list* } } ... }

Specify the contact codes by using the names defined in the technology file.

Via regions for the specified ports use only the specified contact codes. Via regions for all other ports use only the default contact codes. The via regions generated by this option are hard constraints for the signal router.

-preserve_metal_blockage true | **false** | **auto**

Controls whether to preserve all metal shapes and routing blockages around pins as they exist in the design view.

If **true**, all metal shapes and routing blockages around pins are retained exactly. If **false**, the metal shapes and routing blockages are trimmed by pins only when the metal shapes and routing blockages touch a pin. The trimming distance is the maximum spacing value from the technology file. The trimmed metal shapes and routing blockages are converted to zero-spacing routing blockages.

The default value is **auto**. The tool determines whether to preserve the metal shapes and blockages based on the design type of the cell. For library cells, diode cells, end cap cells, fill cells, filler cells, feedthrough cells, and well-tap cells, the tool preserves the metal shapes (**true**). For all other design types, the metal shapes and blockages are trimmed (**false**).

If this option setting conflicts with the setting of the **-trim_metal_blockage_around_pin** option for a layer, the **-trim_metal_blockage_around_pin** option has priority for that layer.

-trim_metal_blockage_around_pin *layer_mode_list*

Specifies how to trim metal shapes and routing blockages around pins by layer. To prevent unpredictable spacing violations during routing, make sure that you set the proper value for this option.

Specify this argument using the following syntax:

{ { *layer_name* touch|all|none } ... }

If set to **touch**, only the metal shapes and routing blockages that touch a pin are trimmed. If set to **all**, all metal shapes and routing blockages around pins are trimmed. If set to **none**, no metal shapes and routing blockages are trimmed. The trimming distance is the maximum spacing value from the technology file. The trimmed metal shapes and routing blockages are converted to zero-spacing routing blockages.

The default is null, and the behavior is controlled by the setting of the **-preserve_metal_blockage** option.

When this option is specified for a layer, its setting has priority over the **-preserve_metal_blockage** setting for that layer.

-read_fill_cells true | **false**

Controls whether to extract geometries in fill cells.

If **true**, the tool performs extraction for fill cells. If **false**, the tool ignores the geometries in fill cells.

The default is **false**.

-remove_non_pin_shapes *layer_mode_list*

Specifies the mode to remove the non-pin shapes by layer.

Specify this argument using the following syntax:

{ { *layer_name* none|all|core } ... }

- **none**
  The non-pin shapes of the given layer are not removed and which is the default behavior.
- **all**
  All non-pin shapes of the given layer are removed.
- **core**
  The non-pin shapes inside the core area of the given layer are removed. The core area is within the DRC distance from the edge of the block boundary.

This option is recommended to use with the **-block_all** or **-block_core_margin** option. The combination can create proper blocked area from the boundary and make sure that there is no missing DRC for the routing wire due to the non-pin shapes are removed.

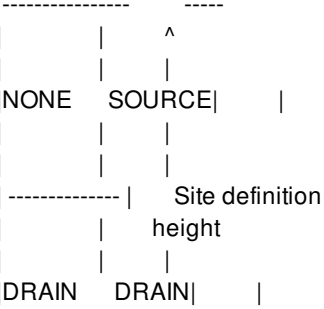
-source_drain_annotation *file_name*

Specifies the name of the input file that contains source-drain annotation information for the cell. The file identifies the type of transistor geometry (SOURCE, DRAIN, or NONE) that abuts each side (left or right) and subrow (upper or lower half of the tile height) of the cell. The tool reads this source-drain information from the file and annotates the information in the frame view. Each side of each subrow of the cell receives one annotation: SOURCE, DRAIN, or NONE; the default annotation is DRAIN. Extraction of this property requires the site definition information.

The following is an example source-drain annotation file:

```
CELL cell_name
  LEFT
    DRAIN ROW1_1
    NONE ROW1_2
  END LEFT
  RIGHT
    DRAIN ROW1_1
    SOURCE ROW1_2
  END RIGHT
END CELL
```

This example annotates a single-height standard cell as follows:



The ROW*x_y* notation means row number *x* and subrow number *y*, counting from bottom to top, starting with 1 and up to the number of rows:

ROW1_1 = row 1, lower half of row 1
ROW1_2 = row 1, upper half of row 1
ROW2_1 = row 2 (upper row of double-height cell), lower half of row 2
ROW2_2 = row 2 (upper row of double-height cell), upper half of row 2
...

DESCRIPTION

This command creates a frame view for the current block by extracting blockage, pin, and via information from the design view. The frame view of a block contains all the information needed for placement and routing. The exclusion of unnecessary data reduces the database size and routing time.

As part of the frame extraction process, the tool determines the region within each pin where the center of a via can connect. This region, called the via region for the pin, is part of the generated frame view. This information gives the router guidance about where it can make connections to a pin. As a result, the router can quickly find the via connection position. Calculating the via region during extraction saves significant runtime for router.

Obstructions in a frame view include three kinds of geometries: shape, regular routing blockage, and zero-spacing routing blockage. A shape is a non-pin geometry on the metal, cut, and device layers defined in the Layer sections of the technology file. A regular routing blockage is a routing blockage with an **is_zero_spacing** attribute of **false**. A zero-spacing routing blockage is a routing blockage with an **is_zero_spacing** attribute of **true**. The router checks all design rules defined in the technology file between the net shapes and the shapes in the frame view. The router treats a regular routing blockage as a shape to keep a safe distance from the surrounding objects. For a zero-spacing routing blockage, the router checks only that there is no overlap (zero spacing) between the blockage and the net shapes.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example extracts the blockage, pin, and via information from the current block.

```
prompt> create_frame
```

The following example uses nondefault via definitions in addition to the default via definitions during via region extraction.

```
prompt> create_frame -include_nondefault_via {VIA23 VIA23A}
```

The following example generates a via region for contact codes VIA12HH and VIA12HV for port A and a via region for contact code VIA12 for port B.

```
prompt> create_frame \
  -port_contact_selections { {A {VIA12HH VIA12HV}} {B {VIA12}} }
```

SEE ALSO

create_abstract(2)

create_freeze_silicon_leq_change_list

Creates logic equivalence (LEQ) change list for engineering change order (ECO) cells. This command is typically used in the freeze silicon ECO flow.

SYNTAX

status **create_freeze_silicon_leq_change_list**
[-cells *cell_objects*]
-output *output_file_name*

Data Types

cell_objects collection
output_file_name string

ARGUMENTS

-cells cell_objects

Specifies the ECO cells for which to create LEQ change list. The given cells must meet three requirements:

- Be leaf cell.
- Have the is_fs_eco_add attribute as true.
- Have the eco_change_status attribute as one of create_cell, add_buffer, add_buffer_on_route, change_link and size_cell.

If this option isn't specified, this command will create LEQ change list for all mappable ECO cells in current design. These ECO cells meet the above mentioned requirements.

-output *output_file_name*

Specifies name of the file to write LEQ change list. If this file doesn't exist, the command will create one. If this file already exists, its contents are discarded and the file is treated as a new empty file.

DESCRIPTION

The **create_freeze_silicon_leq_change_list** command automatically searches for LEQ replacement for ECO cells. It skips same reference lib mapping. For each ECO cells, the command outputs a bunch of TCL commands to the specified file. These TCL commands either rebind ECO cell to a reference library that has instantiated spare cell, or replace ECO cell with its LEQ replacement.

The command first searches for a logic equivalence replacement for each ECO cell one by one. Spare cells in current design meets following requirements can be used to create LEQ change list:

- Its is_eco_fs_dont_use attribute hasn't been defined, or value of is_eco_fs_dont_use is false.
- Its fs_mapped_cell_name attribute hasn't been defined, or value of fs_mapped_cell_name hasn't been set.

Currently this command supports maximum of 2 gate level logic equivalence. This means if the LEQ replacement for a cell has its gate level greater than 2, the command won't create LEQ change list for the cell.

Logic equivalence has two types:

- Same function Id mapping

  If there is a spare cell having same function Id as the ECO cell, this command will reserve the spare cell for same function Id mapping.

- LEQ mapping

  If same function Id mapping fails, this command will try LEQ replacement rules on the cell, until find a LEQ replacement that can be composed of existing spare cells.

Then the command dumps LEQ change list to the output file:

- Same function Id mapping

  This command dumps **size_cell** command to the output file to rebind ECO cell to the reference library of the spare cell that has same function id as the ECO cell.

- LEQ mapping

  This command dumps **remove_cell** command to the output file first to remove ECO cell, then dump a series of **create_cell**, **create_net**, **connect_net** commands to create LEQ.

After running this command, User can check the commands in the output file and then source the file.

EXAMPLES

Suppose there is a ECO cell. Its reference library is AND2X1. Its function is AND. There is a spare cell in current design. Its reference library is AND2XL. AND2XL has same function id as AND2X1. The command will rebind the ECO cell to AND2XL.

```
prompt > create_freeze_silicon_leq_change_list -output leq.tcl
```

leq.tcl will have following content:

```
size_cell ECO_0 AND2XL -not_spare_cell_aware
```

If there is no same function Id spare cell, but there are two spare cells NAND and INV, then leq.tcl will look like:

```
remove_cell ECO_0
create_module AND2X1_leq_1
create_cell ECO_0 AND2X1_leq_1
current_instance ECO_0
create_port -direction in A
create_net A
connect_net A A
create_port -direction in B
```

```
create_net B
connect_net B B
create_port -direction out Y
create_net Y
connect_net Y Y
create_cell leq_cell_0 INVX2
create_net leq_net_1
connect_net leq_net_1 leq_cell_0/A
connect_net Y leq_cell_0/Y
create_cell leq_cell_1 NAND2X1
connect_net A leq_cell_1/A
connect_net B leq_cell_1/B
connect_net leq_net_1 leq_cell_1/Y
```

SEE ALSO

place_freeze_silicon(2)
check_freeze_silicon(2)

create_frontend_tcd_cells

Inserts front-end testkey critical dimension cells (TCDs) into the design.

SYNTAX

status **create_frontend_tcd_cells**
-lib_cells *libcells*
-window_size *{width height}*
[-tcd_spacing *distance*]
[-other_cell_spacing *distance*]
[-icovl_spacing *distance*]
[-respect_blockage none | hard_only | both]
[-density *fvalue*]
[-orientation *orient*]
[-check_only]
[-include_small_windows]
[-bbox *{{x1 y1} {x2 y2}}*]
[-snap_to_fin_grid]
[-place_at_window_center]
[-min_tcd_count *count*]
[-max_tcd_spacing *spacing*]

Data Types

libcells list of lib cells
width float
height float
distance float
fvalue float
orient orientation
x1 float
y1 float
x2 float
y2 float
count int
spacing float

ARGUMENTS

-lib_cells *libcells*

Specifies the front-end TCD library cells.

-window_size ***{width height}***

Specifies the width and height of insertion window size.

-tcd_spacing ***distance***

Specifies the minimum distance between two front-end TCD cells.

-other_cell_spacing ***distance***

Specifies the minimum distance between front-end TCD cells and cells other than front-end TCDs and ICOVL cells.

-icovl_spacing ***distance***

Specifies the spacing between front-end TCDs cells and ICOVL cells.

-respect_blockage none | **hard_only** | **both**

Specifies whether the front-end TCD cell should respect blockages. Possible values are **none**, **hard_only**, and **both** defined as follows:

- none: Ignore all blockages (Default).
- hard_only: Respect hard blockages only.
- both: Respect both hard and soft blockages.

-density ***fvalue***

Specifies the target percentage of windows that contains a front-end TCD cell.

-orientation ***orient***

Specifies the front-end TCD cell orientation. Valid values are R0, R90, R180, R270, MX, MXR90, MY, MYR90.

-check_only

Checks the current design for front-end TCD violations based on rules specified in other options. An error message is issued for each violation.

-include_small_windows

Includes small windows at the last row/column for TCD insertion. By default, these small windows are excluded.

-bbox ***{{x1 y1} {x2 y2}}***

Specifies an insertion area. This area will be divided into the specified window size for TCD insertion. By default, the die area is the insertion area.

-snap_to_fin_grid

Places the lower-left corner of the TCD cell to align with the FinFET grid. By default, alignment is not required.

-place_at_window_center

Changes the placement preference for TCD cells to be close to window center when specified to:

1. Place TCD in center of window

2. Place TCD in the channel between blocks or IPs (if blocks are not abut)

3. Place TCD at blocks or IP corner

By default, TCD cells are not placed at the center of window.

-min_tcd_count *count*

Specifies the minimum number of TCD that must be inserted by this command.

-max_tcd_spacing *spacing*

Specifies the maximum spacing between frontend TCD cells.

DESCRIPTION

This command inserts front-end testkey critical dimension cells (front-end TCDs), which are used to increase design for manufacturability (DFM) and yield.

Front-end TCDs are inserted during floorplanning stage, after macro placement and before tap cells, power switch cells, pre-inserted cells, and so on. Front-end TCDs have no metal layers and no power ports, therefore allowing routing over them.

TCD width and height are not even multiples of unit tile width and height. They are placed similar to regular macros.

A front-end TCD cell is a type of cover cell that allows overlap with other cells.

Front-end TCD cell insertion should prevent die cell area penalty by honoring the following insertion priority:

- Place at window center if -place_at_window_center option is specified
- Place TCD in the channel between blocks or IPs (if blocks are not abut)
- Place TCD in blocks or IP corner
- Place TCD in die area

EXAMPLES

The following example inserts the front-end TCD cell DM1 into windows of width 1000 microns and height 1200 microns. Spacing between DM1 cells is 10 microns. Spacing between DM1 cells and other cells is 2 microns.

```
prompt> create_frontend_tcd_cells -lib_cells {DM1} \
  -window_size {1000 1200} -tcd_spacing 10 -other_cell_spacing 2
```

SEE ALSO

create_backend_tcd_cells(2)
check_tcd_cells(2)

create_generated_clock

Creates a generated clock object.

SYNTAX

collection **create_generated_clock**
[-name *clock_name*]
-source *master_pin*
[-divide_by *divide_factor* | -multiply_by *multiply_factor* |
-edges *edge_list*]
[-combinational]
[-duty_cycle *percent*]
[-invert]
[-preinvert]
[-edge_shift *edge_shift_list*]
[-add]
[-master_clock *clock*]
source_objects
[-comment *comment*]

Data Types

clock_name string
master_pin list
divide_factor int
multiply_factor int
edge_list list
percent float
edge_shift_list list
clock string
source_objects list
comment string

ARGUMENTS

-name *clock_name*

Specifies the name of the generated clock. If you omit this option, the clock receives the same name as the first clock source specified in the **-source** option. If you specify the **-add** option, you must use the **-name** option and the clocks with the same source must have different names.

-source *master_pin*

Specifies the master clock source (a clock source pin in the design) from which to derive the clock waveform. Note that the actual

delays (latency) for the generated clock are computed using its own source pins and not the *master_pin*.

-divide_by *divide_factor*

Specifies the frequency division factor. If the *divide_factor* value is *2*, the generated clock period is twice as long as the master clock period.

-multiply_by *multiply_factor*

Specifies the frequency multiplication factor. If the *multiply_factor* value is *3*, the generated clock period is one-third as long as the master clock period.

-edges *edge_list*

Specifies a list of integers that represents edges from the source clock that are to form the edges of the generated clock. The edges are interpreted as alternating rising and falling edges and each edge must be not less than its previous edge. The number of edges must be an odd number and not less than 3 to make one full clock cycle of the generated clock waveform. For example, *1* represents the first source edge, *2* represents the second source edge, and so on.

-combinational

The source latency paths for this type of generated clock only includes the logic where the master clock propagates. The source latency paths will not flow through sequential element clock pins, transparent latch data pins or the source pins of other generated clocks.

-duty_cycle *percent*

Specifies the duty cycle, as percentage, if frequency multiplication is used. Duty cycle is the high pulse width.

-invert

Inverts the generated clock signal in the case of frequency multiplication and division. This option first creates the gen- erated clock and then inverts the generated clock signal.

-preinvert

Creates a generated clock based on the inverted clock signal. This option first inverts the signal and then creates the gener- ated clock signal.

-edge_shift *edge_shift_list*

Specifies a list of floating point numbers that represents the amount of shift, in library time units, that the specified edges are to undergo to yield the final generated clock waveform. The number of edge shifts specified must be equal to the number of edges specified. The values can be positive or negative. positive indicates a shift later in time, negative a shift earlier. For example, *1* indicates that the corresponding edge is to be shifted by one library time unit.

-add

Specifies whether to add this clock to the existing clock or to overwrite. Use this option to capture the case where multiple generated clocks must be specified on the same source, because multiple clocks fan into the master pin. Ideally, one generated clock must be specified for each clock that fans into the master pin. If you specify this option, you must also use the **-name** and **-master_clock** options.

Defining multiple clocks on the same source pin or port causes longer runtime and higher memory usage than a single clock, because the tool must explore all possible combinations of launch and capture clocks. Use the **set_false_path** command to disable unwanted clock combinations.

-master_clock *clock*

Specifies the master clock to be used for this generated clock if multiple clocks fan into the master pin. If you specify this option, you must also use the **-add** option.

source_objects

Specifies a list of ports, pins or nets defined as generated clock source objects. When a net is used as the source, the first driver pin of the net is the actual source used in creating the generated clock.

-comment ***comment***

Associates a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

DESCRIPTION

Creates a generated clock object. Creates a generated clock in the current design. If successful, the command returns a collection containing the new or modified generated clock.

You can specify a pin or a port as a generated clock object. The command also specifies the clock source from which it is generated. The advantage of using this command is that whenever the master clock changes, the generated clock automatically changes.

The generated clock can be created as a frequency divided clock (by using the **-divide_by** option), frequency multiplied clock (by using the **-multiply_by** option), special divide by one (by using the **-combinational** option), or an edge-derived clock (by using the **-edges** option). The frequency-divided or frequency-multiplied clock can be inverted by using the **-invert** option. The shifting of edges of the edge-derived clock is specified by using the **-edge_shift** option. The **-edge_shift** option is used for intentional edge shifts and not for clock latency.

The number of edges specified by **-edges** to make one period of the generated clock waveform must be an odd number equal to or greater than 3. For example, in the following command:

create_generated_clock -source clk **-edges** { 1 3 5 } [get_pins flop/Q]

edge 1 indicates the first rising edge of the generated clock, edge 3 indicates the first falling edge of the generated clock, and edge 5 indicates the next rising of the generated clock. Note that the period of the generated clock is determined by the final entry in the edge list.

Non-increasing edges, such as **-edges { 1 1 3 }**, are allowed and usually used with **-edge_shift** to produce a generated clock pulse independent of the duty cycle of the master clock itself.

If a generated clock is specified with a *divide_factor* value that is a power of 2 (1, 2, 4, ...), the rising edges of the master clock are used to determine the edges of the generated clock. If the *divide_factor* value is not a power of two, the edges are scaled from the master clock edges.

Using the **create_generated_clock** command on an existing *generated_clock* object overwrites its attributes. The *generated_clock* objects are expanded to real clocks at the time of analysis.

The following commands can reference the generated_clock: **set_clock_latency**, **set_clock_uncertainty**, **set_propagated_clock**, and **set_clock_transition**.

For internally generated clocks, the tool automatically computes the clock source latency if the master clock of the generated clock has propagated latency and no user-specified value for generated clock source latency exists.

If the master clock is ideal and has source latency, and there is no user-specified value for the generated clock's source latency, then zero source latency is assumed.

To display information about generated clocks, use the **report_clocks** command.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example creates a frequency divide_by 2 generated clock.

```
prompt> create_generated_clock -divide_by 2 \
  -source [get_pins CLK] [get_pins u0]
```

The following example creates a frequency divide_by *3* generated_clock. If the master clock period is *30*, and master waveform is *{24 36}*, the generated clock period is *90* with waveform *{72 108}*.

```
prompt> create_generated_clock -divide_by 3 \
  -source [get_pins CLK] [get_pins div3/Q]
```

The following example creates a frequency multiply_by *2* generated_clock with a duty cycle of *60*%.

```
prompt> create_generated_clock -multiply_by 2 -duty_cycle 60 \
  -source [get_pins CLK] [get_pins u1]
```

The following example creates a frequency multiply_by *3* generated_clock with a duty cycle equal to the master clock duty cycle. If the master clock period is *30*, and master waveform is *{24 36}*, the generated clock period will be *10* with waveform *{8 12}*.

```
prompt> create_generated_clock -multiply_by 3 \
  -source [get_pins CLK] [get_pins div3/Q]
```

The following example creates a generated clock whose edges are edges *1*, *3*, and *5* of the master clock source.

```
prompt> create_generated_clock -edges {1 3 5} \
  -source [get_pins CLK] [get_pins u2]
```

The following example shows the generated clock in the previous example with each derived edge shifted by *1* time unit.

```
prompt> create_generated_clock -edges {1 3 5} -edge_shift {1 1 1} \
  -source [get_pins CLK] [get_pins u3]
```

The following example creates an inverted clock.

```
prompt> create_generated_clock -divide_by 1 -invert
```

The following example creates a rising edge pulse triggered by the rising edge of its master clock.

```
prompt> create_generated_clock -edges {1 1 3} -edge_shift {0 5 0} \
  -source [get_pins CLK] [get_pins u4]
```

The following example creates a falling edge pulse triggered by the rising edge of its master clock with a period 10.

```
prompt> create_generated_clock -edges {1 1 3} -edge_shift {0 5 0} \
  -invert -source [get_pins CLK] [get_pins u5]
```

The following example creates a frequency doubling pulse. A rising edge pulse triggered by both rising and falling edges of its master clock with a period of 10.

```
prompt> create_generated_clock -edges {1 1 2 2 3} -edge_shift {0 2.5 0 2.5 0} \
  -source [get_pins CLK] [get_pins u6]
```

SEE ALSO

create_clock(2)
get_generated_clocks(2)
remove_generated_clocks(2)
report_clocks(2)
report_timing(2)
set_clock_latency(2)
set_clock_transition(2)
set_clock_uncertainty(2)
set_propagated_clock(2)

create_geo_mask

Creates a geo_mask object.

SYNTAX

collection **create_geo_mask**
[-objects *object_list*]
[*pos_object_list*]
[-merge]

Data Types

object_list collection
pos_object_list collection

ARGUMENTS

-objects *object_list*

Specifies the objects to be used to define the geometric region to be covered by the geo_mask. Objects may be a heterogenous collection of poly_rects, geo_masks, shapes, layers, and other physical objects.

In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

pos_object_list

Specifies the objects to be used to define the geometric region to be covered by the geo_mask. This positional option is provided for compatibility with other tools.

-merge

Indicates whether the resulting geo_mask should be merged. By default, the resulting geo_mask may contain distinct overlapping or adjacent regions. When this flag is set, the area in the geo_mask is simplified by removing overlapping area and merging adjacent regions.

DESCRIPTION

This command creates a new geo_mask object and returns a collection that contains the new object. geo_masks represent polygonal regions of arbitrary angularity, and can be used for geometrical computation. The region represented by a geo_mask may

not necessarily be contiguous, and may also be empty. Calling this command with no arguments will return a geo_mask representing an empty region with zero area.

EXAMPLES

The following example creates a geo_mask with a rectangular region.

```
prompt> create_geo_mask -objects [create_poly_rect -boundary {{0 0} {200 100}}]
```

The following example creates a geo_mask with a region defined by shapes on a layer, merged.

```
prompt> create_geo_mask [get_layers M1] -merge
```

SEE ALSO

copy_to_layer(2)
create_poly_rect(2)
compute_polygons(2)
resize_polygons(2)
split_polygons(2)
compute_area(2)
transform_polygons(2)

create_grid

Creates a new user or block grid.

SYNTAX

collection **create_grid**
[-pg_strategy *PG_strategy*]
[-layers *layers*]
[-site_rows *rows*]
[-site_arrays *arrays*]
[-x_step *step*]
[-y_step *step*]
[-x_offset *offset*]
[-y_offset *offset*]
[-orientations *allowed_orientations*]
[-type block | user]
grid_name

Data Types

PG_strategy	string
layers	collection
rows	collection
arrays	collection
step	float
offset	float
allowed_orientations	list
grid_name	string

ARGUMENTS

-pg_strategy *PG_strategy*

Specifies the name of the PG strategy to use when creating the block grid. This option is applicable only for auto-derived block grid creation. The tool derives the grid pitches and offsets from the PG strategy. It is mutually exclusive with any option for manual block grid.

-layers *layers*

Specifies the tech layers to consider within the PG strategy. This option is applicable only for auto-derived block grid creation. A PG strategy might involve many routing layers. By default, all the layers in the specified PG strategy are used to derive the grid. If *-layers* is specified, only the specified layers in the PG strategy are used. This option is mutually exclusive with any option for manual block grid.

-site_rows *rows*

Specifies the site rows to be considered in creating the grid. This option is applicable only for auto-derived block grid creation. The tool derives the grid's pitches and offsets from the given site rows. The specified site rows must be uniform and of the same type of unit site def. This option is mutually exclusive with any option for manual block grid.

-site_arrays ***arrays***

Specifies the site arrays to be considered in creating the grid. This option is applicable only for auto-derived block grid creation. The tool derives the grid's pitches and offsets from the given site arrays. This option is mutually exclusive with any option for manual block grid.

-x_step ***step***

Specifies the grid pitch in the x-direction. The default pitch is the litho grid pitch of the design. This option is applicable for both user grid and block grid creation. This is an option for manual block grid and is mutually exclusive with any option for auto-derived block grid.

-y_step ***step***

Specifies the grid pitch in the y-direction. The default pitch is the litho grid pitch of the design. This option is applicable for both user grid and block grid creation. This is an option for manual block grid and is mutually exclusive with any option for auto-derived block grid.

-x_offset ***offset***

Specifies the grid offset from the design origin in the x-direction. The default offset is 0. This option is applicable for both user grid and block grid creation. This is an option for manual block grid and is mutually exclusive with any option for auto-derived block grid.

-y_offset ***offset***

Specifies the grid's offset from the design origin in the y-direction. The default offset is 0. This option is applicable for both user grid and block grid creation. This is an option for manual block grid and is mutually exclusive with any option for auto-derived block grid.

-orientations ***allowed_orientations***

Specifies the allowed orientations for any associated block instance of the created block grid. This option is applicable only for block grid creation. Specifies one or more of: R0, MX, MY, R180. By default, all four orientations are allowed. This is an option for manual block grid and is mutually exclusive with any option for auto-derived block grid. The allowed orientations of an auto-derived block grid are derived automatically from the PG strategy, layer, and site rows.

-type block | **user**

Specifies the type of grid to be created. By default a block grid is created. The **-type** option has two possible values: *block* for block grid, or *user* for user grid.

grid_name

Specifies the name of the grid to be created.

DESCRIPTION

This command creates a new grid. Two grid types are supported: block and user. The block grid can be used to tune block locations with commands such as *shape_blocks*, or by editing the layout in the GUI.

Use the **set_block_grid_references** command to associate the grid with a block reference design. Snapping the block instances to the block grid is done automatically inside the **shape_blocks** command and when editing the layout in the GUI with the snap

setting. You can also use the **snap_cells_to_block_grid** command to snap block instances to the block grid.

EXAMPLES

The following example creates an auto-derived block grid named gr1 from patterns on layers M4 and M5 of PG strategy s1.

```
prompt> create_grid -pg_strategy s1 -layers [get_layers {M4 M5}] gr1
```

The following example creates manual block grid named gr2 with pitch of 20 microns in the y-direction. All associated block instances must be placed in orientation R0, or flipped around the y-axis.

```
prompt> create_grid -y_step 20 -orientations {R0 MY} gr2
```

The following example creates a user grid named ug1 with a pitch of 10 microns in both x- and y-directions at an offset of 12 microns in the x-direction.

```
prompt> create_grid -type user -x_step 10 -y_step 10 -x_offset 12 ug1
```

SEE ALSO

get_grids(2)
remove_grids(2)
report_grids(2)
set_block_grid_references(2)
set_grid(2)
snap_cells_to_block_grid(2)

create_group

Creates a group object in the current design.

SYNTAX

collection **create_group**
[-name *group_name*]
[-allow_duplicate_names]
[-type collection | set]
[-remove_when any_member_removed | empty | no_auto_removal | one_member]
[-shaping]
[*object_list*]

Data Types

group_name string
object_list list

ARGUMENTS

-name *group_name*

Specifies the name of the group. By default, the command generates a name automatically and uses the prefix GROUP_0, GROUP_1, and so on for new groups.

-allow_duplicate_names

Allows duplicate group names in the design. When a new group is created with the **-allow_duplicate_names** option, the tool allows other groups to be created with the same name.

By default, the tool ensures that group names are unique in the design. When a new group is created, the tool validates that no other group has the same name and ensures that no other group is ever assigned the same name.

-type collection | set

Sets the type of the group. By default, the type is *set*.

A member of the group can appear only one time in a *set* type. A member of the group can appear multiple times in a *collection* type group.

-remove_when any_member_removed | empty | no_auto_removal | one_member

Specifies when the group should be automatically removed from the design, based on the removal of member objects in the group.

- any_member_removed: Remove the group when any member object is removed from the group.

- empty: Remove the group when the last member from the group is removed.
- one_member: Remove the group when only one member is left in the group after removing a different member.
- no_auto_removal: Never remove the group automatically.

By default, the *no_auto_removal* argument is used.

-shaping

The -shaping argument creates the group as a shaping group. Shaping groups must be sets and may only contain shapeable objects (voltage areas, move bouds, cells, and shaping groups). This argument is mutually exclusive with -repelling.

-repelling

The -repelling argument creates the group as a repelling bound group. Repelling bound groups must be sets and remove when empty. Furthermore, they may only contain cells. This argument is mutually exclusive with -shaping.

object_list

Specifies the objects to be added to the group.

DESCRIPTION

The **create_group** command creates a group object. You can use a group object to implement a general purpose collection of design scoped supported objects. You can bundle these design scoped objects in a group and then can use as per the requirement.

Objects from the same design can be added to groups with the **add_to_group** command, and removed with the **remove_from_group** command. An object can belong to any number of groups at a time. Removing an object from a group does not remove the object from the design, however removing an object from the design automatically removes it from its containing group. Group nesting is permitted, so groups can contain other groups as member object, however cyclic relationship is not permitted between two groups so two groups cannot be member to each other.

This command returns the group as a single element collection, an empty string if it fails, or a Tcl error if there is a command syntax error.

EXAMPLES

The following example creates a group named "group1", which is automatically removed when its first member object is removed.

```
prompt> create_group -name "group1" -remove_when any_member_removed
{group1}
```

The following example creates a group named "group2" and adds the "CornUR" cell to the group.

```
prompt> create_group -name group2 [get_cells CornUR]
{group2}
```

SEE ALSO

add_to_group(2)
get_groups(2)
remove_from_group(2)
remove_groups(2)
report_groups(2)

create_group_repeaters_guidance

Create net groups using smart net grouping algorithms.

SYNTAX

int **create_group_repeaters_guidance**
-nets *collection_of_nets*
[-group_space *site_height_multiplier*]
[-exclude_pin_ports]
[-ignore_layers]

Data Types

collection_of_nets	collection
site_height_multiplier	float

ARGUMENTS

-nets *collection_of_nets*

Sepcifies multiple nets for adding group repeaters. This option is required.

-group_space *site_height_multiplier*

Specified that repeater groups can be inserted on top of blocks. The default value is 2.0. This is optional.

-exclude_pin_ports

Driver and load pins are ignored when grouping nets. With this option, signal direction in a net group may not be the same.

-ignore_layers

Routing layers are ignored when grouping nets. With this option, nets in the same group may not route on the same layer.

DESCRIPTION

This command creates net groups based on their routing topology, signal direction, and routing layers. The net group IDs are set on net attribute *eco_net_group_id*.

EXAMPLES

The following examples show usages of command options.

prompt> **create_group_repeaters_guidance -nets nets**

prompt> **create_group_repeaters_guidance -nets -group_space 5 -exclude_pin_port**

SEE ALSO

add_group_repeaters(2)

create_hdl2upf_vct

Defines a value conversion table that can be used to convert HDL logic values into UPF net state type values. This command is supported only in UPF mode.

SYNTAX

status **create_hdl2upf_vct**
vct_name
-hdl_type {vhdl|sv [*typename*]}
-table {{*from_value to_value*}*}

Data Types

vct_name string
typename string
from_value HDL logic value
to_value UPF state type value

ARGUMENTS

vct_name

The value conversion table name.

-hdl_type {vhdl|sv *typename*}

The HDL type for which the value conversions are defined.

-table {{*from_value to_value*}*}

A list of the values of the HDL type to map to UPF state type values.

DESCRIPTION

The **create_hdl2upf_vct** command defines a value conversion table from an HDL logic type to the state type of the supply net value when that value is propagated from HDL port to a UPF supply net. It shall provide a conversion for each possible logic value that the HDL port can have. The **create_hdl2upf_vct** command does not check that the set of HDL values are complete or compatible with any HDL port type.

The *vct_name* argument provides a name for the value conversion table for later use with the **connect_supply_net** command (see 6.6).

The **-hdl_type** argument specifies the HDL type for which the value conversions are defined. This information allows a tool to provide completeness and compatibility checks. If the typename is not one of the language's predefined types or one of the types specified in the next paragraph, then it shall be of the form library.pkg.type.

The following HDL types shall be the minimum set of types supported. An implementation tool may support additional HDL types:

-) VHDL
  - Bit, std_[u]logic, Boolean
  - Subtypes of std_[u]logic
- SystemVerilog
  - reg/wire, Bit, Logic

The **-table** option defines the 1:1 conversion from HDL logic value to the UPF partially on and on/off states. The values are consistent with the HDL type values.

For example:

- When converting from SystemVerilog logic type, the legal values are 0, 1, X, and Z.
- When converting from SystemVerilog or VHDL bit, the legal values are 0 or 1.
- When converting from VHDL std_[u]logic, the legal values are U, X, 0, 1, Z, W, L, H, and -.

The conversion values have no semantic meaning in UPF. The meaning of the conversion value is relevant to the power-aware HDL model to which the supply net is connected.

SEE ALSO

create_upf2hdl_vct(2)
connect_supply_net(2)

create_icovl_cells

Inserts in-chip overlap (ICOVL) cells for layer-to-layer alignment control.

SYNTAX

status **create_icovl_cells**
-lib_cells *libcells*
[-partitions *inst_pair*]
[-xy_grid *int_pair*]
[-type backend | frontend]
[-tcd_spacing *distance*]
[-other_cell_spacing *distance*]
[-icovl_spacing *distance*]
[-orientation R0 | R90 | R180 | R270 | MX | MXR90 | MY | MYR90
[-check_only]
[-bbox *{{x1 y1} {x2 y2}}*]
[-placement_blockage_extension *extension*]
[-routing_blockage_extension *extension*]
[-routing_blockage_layers *{start_layer end_layer}*]
[-over_icovl_routing_guide_extension *extension*]
[-over_icovl_routing_guide_layers *{start_layer end_layer }*]

Data Types

int_pair 2 integers
libcells list of library cells
inst_pair list of instance pairs
distance float
x1 float
y1 float
x2 float
y2 float
extension float
start_layer layer
end_layer layer

ARGUMENTS

-xy_grid *int_pair*

Divide the chip area into boxes based on specified grids. Insert one icovl cell to each grid box if possible. Either us -xy_grid or -partitions option to specify how icovl cells should be inserted

-lib_cells *libcells*

Specifies the collection of ICOVL library cells to insert as a set.

-partitions *inst_pair*

Specifies a list of integer pairs that describe how the insertion area should be divided and specify the number of ICOVL sets that should be inserted. The first number of each integer pair is the relative size of the insertion area. The second number of each pair is the number of ICOVL sets for this part. For example, the following option

-partitions {{1 1} {7 5} {2 2}}

specifies that insertion area should be vertically divided into 1+7+2=10 parts. The first partition is equal to 1 part and should contain one set of ICOVL cells. The second partition is equal to 7 parts and should contain 5 sets of ICOVL cells. The third partition is equal to 2 parts and should contain 2 sets of ICOVL cells.

-type backend | **frontend**

Specifies the type of ICOVL insertion. Valid values are *frontend* or *backend*. Use this option to indicate you are doing frontend or backend ICOVL insertion.

-tcd_spacing *distance*

Specifies the minimum distance between ICOVL and testkey critical dimension (TCD) cells.

-other_cell_spacing *distance*

Specifies the minimum distance between ICOVL cells and cells other than testkey critical dimension (TCD) and ICOVL cells.

-icovl_spacing *distance*

Specifies the spacing between ICOVL cells.

-orientation R0 | **R90** | **R180** | **R270** | **MX** | **MXR90** | **MY** | **MYR90**

Specifies the front-end ICOVL cell orientation. Valid values are R0, R90, R180, R270, MX, MXR90, MY, MYR90.

-check_only

Checks the current design for front-end ICOVL violations based on rules specified in other options. An error message is issued for each violation.

-bbox *{{x1 y1} {x2 y2}}*

Specifies an insertion area in which to insert the ICOVL cells. This area will be divided into the specified window size for ICOVL insertion. By default, the die area is the insertion area.

-placement_blockage_extension *extension*

Specifies the extension size of placement blockage for ICOVL cells in microns. The blockage will extend beyond ICOVL boundary by the size specified. When specified, the tool creates placement blockages around new ICOVL cells. Note that this placement blockage does not prevent existing cells from placement within the blockage. To keep other cells outside the placement blockage, apply a cell spacing rule that is greater or equal to placement blockage size. By default, no placement blockage extension is applied.

-routing_blockage_extension *extension*

Specifies the extension size of routing blockage for ICOVL cells in microns. The blockage will extend beyond ICOVL boundary by the specified size. Routing blockages will be created on layers specified in **-routing_blockage_layers**. You must specify the **-routing_blockage_layers** option together with the **-routing_blockage_extension** option. This option is not required.

-routing_blockage_layers *{start_layer end_layer}*

Specifies the metal routing blockage layers. Routing blockages will be created on each metal layer from *start_layer* to *end_layer*. To specify only one routing blockage layer, use the same layer name for both *start_layer* and *end_layer*, for example, **-routing_blockage_layers {M3 M3}**. This option is not required.

-over_icovl_routing_guide_extension *extension*

Specifies the size (in micron) for special over-ICOVL cell route guides to be created on ICOVL cells. Over-ICOVL routing guides are created over ICOVL cells and extend outward from an ICOVL cell edge by the specified size. This option must be used together with the **-over_icovl_routing_guide_layers** option. This option is not required.

-over_icovl_routing_guide_layers *{start_layer end_layer }*

Specifies the metal layers for special over-ICOVL cell routing guides to be created on ICOVL cells. To specify a single metal layer for the routing guide, use the same layer name for both start_layer and end_layer, for example, **-over_icovl_routing_guide_layers {M3 M3}**. This option is not required.

DESCRIPTION

This command inserts in-chip overlap (ICOVL) cells into the current design. ICOVL cells are used for layer-to-layer alignment control during fabrication to improve yield. ICOVL cells are included on one layer with different DP masks. ICOVL cells are inserted into the core area of the design. The core area is divided into partitions, where each partition can have a different size and require a different number of ICOVL cell sets. ICOVL cells are macros and are not allowed to overlap any other cells.

This command is used for both frontend and backend ICOVL cell insertion. Frontend ICOVL cells are inserted during the floorplanning stage, after macro and frontend TCD cell placement. Backend ICOVL cells are inserted after power planning.

EXAMPLES

The following command inserts frontend ICOVL cell sets. Each ICOVL sets has 3 ICOVL cells: icovl_m1, icovl_m2, and icovl_m3. The core area is divided into 4 partitions in the y-direction. From bottom to top, the first partition equals 10 percent of the core area and needs 1 ICOVL sets. The 2nd partition equals 30 percent and needs 2 ICOVL sets. The 3rd partition is 20 percent and needs 2 ICOVL sets, and 4th partition is 40 percent and needs 3 ICOVL sets. Routing blockages will be created on metal layers from metal1 through metal6. ICOVL cells will be placed in R0 orientation.

```
prompt> create_icovl_cells -lib_cells {icovl_m1 icovl_m2 icovl_m3} \
  -partitions { {1 1} {3 2} {2 2} {4 3}} \
  -placement_blockage_extension 20.0 \
  -routing_blockage_extension 3.0 \
  -routing_blockage_layers { metal1 metal6} \
  -icovl_spacing 10.0 \
  -tcd_spacing 5.0 \
  -other_cell_spacing 5.0 \
  -orientation R0 \
  -type frontend
```

Below example will divide chip area into 4 x 5 grids and insert one icovl cell into each grid if possible.

```
prompt> create_icovl_cells \
-lib_cells ICOVL_CELL  \
-tcd_spacing 90 \
-xy_grid { 4 5}
```

SEE ALSO

create_backend_tcd_cells(2)
create_frontend_tcd_cells(2)

create_interior_tap_walls

Creates a tap wall inside a boundary edge of a core area, hard macro, soft macro, hard placement blockage, or nondefault voltage area. The boundary typically already contains boundary cells.

SYNTAX

status **create_interior_tap_walls**
-lib_cell *cell_name*
-side top | bottom | left | right
[-x_spacing *spacing*]
[-bbox *coordinates*
[-prefix *cell_prefix*]
[-orientation R0 | R180 | MX | MY]

Data Types

cell_name string
spacing float
coordinates list
cell_prefix string

ARGUMENTS

-lib_cell *cell_name*

Specifies the library reference cell to be used as a tap cell. You must specify a single library cell. This is a required option.

-side top | bottom | left | right

Specifies the side on which to create a tap wall. You must specify a single side. This is a required option.

-x_spacing *spacing*

Specifies the spacing between tap cells in microns for horizontal tap walls. This option is ignored for vertical tap walls.

If the specified value is not an integer multiple of the site width, the tool rounds it down to a multiple of the site width. If the value is less than the width of the tap cell, the tool uses zero spacing.

The default is 0, and there is no space between the tap cells.

-bbox *coordinates*

Specifies the rectangular tap insertion region in microns. Specify the rectangle as a list of four coordinates, {{llx lly} {urx ury}}, which represents the lower-left and upper-right corners of the rectangle.

A tap wall is created along the longest edge of the core boundary, macro, hard placement blockage, or nondefault voltage area

encompassed by the rectangular region. The command issues an error message if it cannot find an edge.

If you do not specify this option, the command creates an interior tap wall along each edge of the specified side.

-prefix ***cell_prefix***

Specifies the prefix for the created tap cells.

When you use this option, the command uses the following naming convention for the inserted tap cells:

*<cell_prefix>__<lib_cell>*_R#_C#_*number*

By default, no prefix is added and the tool uses the following naming convention for the inserted tap cells:

tapfiller__*<lib_cell>*_R#_C#_*number*

-orientation R0 | **R180** | **MX** | **MY**

Specifies the orientation of the placement of the tap cells.

By default, the orientation follows the orientation of the row or site.

DESCRIPTION

This command adds an interior tap wall to the design. A tap wall is a row or column of identical tap cells placed linearly. A tap cell is a special cell with a well tie, substrate tie, or both. Tap cells are typically used when most or all standard cells in the library do not contain substrate or well taps.

Interior tap wall placement follows the usual rules of standard cell placement and snaps the tap cells to rows and sites. The command does not insert tap cells if the rows or sites are missing.

If you specify the **-bbox** option, the command creates a single tap wall along the longest edge segment. If the edge segment does not contain a corner, the tap wall aligns to the left or bottom of the tap insertion region. If the edge segment contains one corner, the tap wall aligns to the corner. If the edge segment contains two corners, the tap wall aligns to the left or bottom of the tap insertion region. When the edge segment contains two corners for a horizontal wall, the gap between the last cells the might be shortened.

Obstructions such as placement blockages, fixed cells, and macros might prevent successful placement of the tap cells, resulting in incomplete or missing tap walls.

The command marks the inserted tap cells as fixed.

EXAMPLES

The following example creates tap walls along the inside of all edges on the top side of the core area.

prompt> **create_interior_tap_walls -lib_cell myLib/Cell1 -side top**

SEE ALSO

create_dense_tap_cells(2)
create_exterior_tap_walls(2)
create_tap_cells(2)
create_tap_meshes(2)

create_interposer_routeplan

Places vias and routes for an interposer design.

SYNTAX

string **create_interposer_routeplan**
[-interposer_style default | fpga | single_hbm | vertical_routing]
[-pattern_file *filename*]

Data Types

filename string

ARGUMENTS

-interposer_style default | fpga | single_hbm | vertical_routing

Specifies the style of the design. Based on the selection, the tool will process different types of pattern files and perform different actions. If this option is not specified, the **default** style is used.

-pattern_file *filename*

Specifies a pattern file that contains information on how to place interposer vias and routes. The keywords supported in the pattern file depend on the **-interposer_style** option specified.

The pattern file contains one or more sections, where each section starts with a keyword followed by a pair of curly brackets. Inside the curly brackets, one or more constraints is specified. Each constraint is made of a pair of curly brackets which contains the information of the element. Comment lines begin with the pound sign (#) and inline comments begin with two forward slashes (//).

If the **default** style is chosen, the following sections and elements are supported:

The configuration section is specified by the *config* keyword and contains a **stack_level** specification and a **via_def** specification. The **stack_level** keyword is followed by a positive integer, where zero specifies that simple vias will be inserted and a positive integer specifies an additional number of stacked via levels. For example, {stack_level 1} specifies that the tool places the basic via and one additional level.

The **via_def** keyword is followed by a non-negative integer and a string. The integer represents the via level in the via stack and the string specifies the via reference, where the specified via reference is used to create new vias. If the integer following the via_def keyword conflicts with the level specified by stack_level, it will be ignored and the tool will issue a warning message. An example configuration section is given below which specifies a stacked via.

```
config {
  {stack_level 2}
  {via_def 0 via7}
  {via_def 1 via6}
```

```
  {via_def 2 via5}
}
```

In the config section, user can describe via arrays. Specifically, the orientation keyword specifies the orientation of the via w.r.t the via_def. It is followed by a string $ORIENTATION describing the orientation. The $bump_instance_name is optional. If it is used, the via configuration is only for the bump cell instance specified. Otherwise, it is the default and will be used for all bumps without individual via configuration information. The columns and rows keyword define the dimension of the via array. They are followed by positive integers. Similarly, they can have the optional $bump_instance_name.

```
config {
{orientation $ORIENTATION  [$bump_instance_name]}
{columns $positive_integer [$bump_instance_name]}
{rows $positive_integer [$bump_instance_name]}
}
```

It is important to note that, if a bump already has a via placed, which has a different configuration as that described by the pattern file, the existing via will not be changed. It is user's responsibility to remove the existing via before running the command with correct via configuration information in the pattern file.

The design rule section, specified with the *drc* keyword, contains spacing rules for via insertion and placement. This section contains the **bump**, **peer**, **layer**, **via_def_name**, and **via_def_index** specifications. The **bump** keyword is followed by a floating point number that specifies the minimum distance between the base via and any bump in microns. The **peer** keyword is followed by a floating point number and specifies the minimum distance between the base vias. The **layer** keyword is followed by a layer name string and a floating point number, and specifies the minimum distance between the base via and a preroute on the specified layer. Only the top and bottom layers of the base via are relevant, all other layers are ignored. The **via_def_name** keyword is followed by a string and a floating point number, and specifies the minimum distance between the base via and the specified type of via. If the specified via def is the base via, this element provides the same functionality as that of the second type element. Otherwise, since the via on any lower metal level is inside the base via, the distance is measured between the edges of the vias inside of the base via. The **via_def_index** keyword is followed by non-negative integer and a floating point number, provides similar functionality to **via_def_name**, but specifies the via by using a level index instead of a via_def name. The base via is on level 0, lower level vias beneath the base via will have indexes of positive values. An example of the design rule section is given below.

```
drc {
  {bump 0.1}
  {peer 0.1}
  {layer AP 0.1}
  {via_def_name VIA34 0.2}
  {via_def_index 1 0.2}
}
```

The via count section, specified with the *count* keyword, contains the rule for the number of vias or stacked vias to add to the interposer. The *bump* keyword specifies that a base via might be added for each front-side bump of a net. The *flat_net* keyword specifies that at most a single base via can be added for the entire flat net. If any part of the flat net has been routed on the layer of the bump, the via location will be selected on a preroute. Otherwise, a bump will be selected for via insertion. The third possible keyword is *bump_share*. If this mode is selected, tool will try to insert an interposer via for each bump of the net. If at least one bump is added for the net, no error message will be issued. Namely, bumps can share an interposer via. If no via can be inserted, an error message will be issued. Note that the vias do not include existing ones. If there are interposer vias already in the design for all bumps of the net, no error message will be issued. If this section is not given, the default value is *bump*. An example of the via count section is shown below.

```
count {
  flat_net
}
```

The exclusion section, specified with the *exclusion* keyword, specifies that specific bump cells, nets, or reference cells should be ignored. The *bump* keyword is followed by a bump instance string. The specified bump cell instance is ignored. The *net* keyword is followed by a flat net string. The specified net is ignored. The *reference_cell* keyword is followed by a reference cell string. All

bump instances of the specified reference cell are ignored for via insertion. By default, the tool processes the entire interposer design. An example of the exclusion section is given below.

```
exclusion {
 {bump BUMP_1}
 {net NET}
 {reference_cell REF_BUMP}
}
```

The inclusion section, specified with the *inclusion* keyword, specifies the part of netlist which the tool will process. Its format is identical to the exclusion section. All three keywords reference_cell, bump, net can be used. The inclusion and exclusion sections are mutually exclusive. An example of the inclusion section is given below.

```
inclusion {
 {bump BUMP_1}
 {net NET}
 {reference_cell REF_BUMP}
}
```

The location section, specified with the *location* keyword, contains location constraints for the new vias. This section contains the **x**, **y**, or **xy** keywords, specifying a horizontal coordinate, a vertical coordinate, both coordinates. The keyword is followed by the **relative** or **absolute** keyword. The **relative** keyword specifies values with respect to the center of the corresponding bump boundary box. The **absolute** keyword specifies values with respect to the interposer origin. The next field specifies one or two float values for the horizontal, vertical, or both coordinates. The optional last field specifies the bump instance name to use when applying the constraint. If there is no bump name, the constraint is applied to all bumps. An example of the location section is given below.

```
location {
 {x  relative 4      BUMP_1}
 {y  absolute 400    BUMP_2}
 {xy relative 2   2  BUMP_3}
}
```

It is important to note that if you specify both horizontal and vertical coordinates for a bump, the tool places a via at the specified location as long as there is not a via already at the exact same location. Therefore, if you run the tool multiple times with different horizontal and vertical coordinates, many vias will be placed for the same bump. It is your responsibility to remove any redundant vias.

The exception section specified with the *exception* keyword, contains many special constraints. If the *default* design style is chosen, user can use **45degree** make the tool place a via close to a corner of the corresponding bump boundary box. Note that this feature is only applied when the bump is at a corner of the correspond net bounding box. Vias placed on nets will not be affected. Using this option will reduce the overall wirelength of the interposer routing.

```
exception {
 {45degree}
}
```

If the *default* design style is chosen, the tool only places interposer vias within the bounding box of the corresponding net. No vias can be placed if there is no enough area for a single via inside the bounding box. User can use **box_enlarge_value** in the exception section specified with the *exception* keyword in the pattern file to increase the area where interposer vias can be placed. Specifically, both dimension of the net bounding box is increased by the amount specified by *box_enlarge_value*. User can use **box_dimension_threshold** in addition to **box_enlarge_value**. If **box_dimension_threshold** is used, only any net bounding box whose dimension is less than the value specified will be increased. The following example enlarges all net bounding box areas whose dimension is less than 10 micron by 0.5 micron.

```
exception {
 {box_dimension_threshold 10}
 {box_enlarge_value 0.5}
}
```

The tool is able to report net routing order by generating text files containing sets of net names. User needs to use the keyword **hbm_report_shielding_groups** followed by net patterns. Only two patterns are supported as shown below. Nets routed in different layers will be grouped in separate files. The filenames are upper_$index and lower_$index, corresponding to the two routing metal layers.

{hbm_report_shielding_groups interleave_4} {hbm_report_shielding_groups interleave_2}

If the **single_hbm** style is chosen, the following elements are supported. They are all inside the exception section. The keyword hbm_channel_direction describes the channel orientation.

{hbm_channel_direction vertical | horizontal}

The routing width and spacing in the bump array region of the HBM channel are specified by the keyword hbm_wire_width and hbm_wire_spacing, respectively. The routing width and spacing in the middle section of the channel are the same unless they are changed by constraints of other keywords. The values following the keywords must be positive. User is responsible for the correctness of the values.

{hbm_wire_width $micron} {hbm_wire_spacing $micron}

The keyword **hbm_via_bump_abutment** is a boolean constraint that determines the routing style produced by the tool. When it is set to true, all vias that connect the top RDL layer to any lower metal layer are placed with bump abutment. As a result, there is no need for RDL routing between vias and bumps. When **hbm_via_bump_abutment** is true, the Manhattan double-Z style is chosen for the routing in the middle section of the HBM channel. The following keywords are used to specifies the route widths and lengths for the middle section routing. Particularly, the middle segment length is described by keyword hbm_midtrack_lengths. The routing width and spacing for the two routing segments perpendicular to the channel orientation are described by keyword hbm_midtrack_turn_width and hbm_midtrack_turn_spacing, respectively. The routing width and spacing for the middle section routing are described by keyword hbm_midtrack_width and hbm_midtrack_spacing, respectively.

{hbm_midtrack_lengths $micron} {hbm_midtrack_turn_spacing $micron} {hbm_midtrack_turn_width $micron} {hbm_midtrack_spacing $micron} {hbm_midtrack_width $micron}

If keyword **hbm_via_bump_abutment** is false, the tool may not place vias with bump abutment. The vias are aligned to the corresponding routing. The gap between the via and bump bounding box is specified by the keyword hbm_rdl_via_offset. The vias are always placed above or on the right side of the corresponding bumps, for vertical and horizontal channels, respectively. When keyword **hbm_via_bump_abutment** is false, the 45-degree Z-style is chosen for the routing in the middle section of the HBM channel. The keyword hbm_middle_turn_offset specifies the length of top or right segment routing, for vertical and horizontal channels, respectively.

{hbm_rdl_via_offset $micron} {hbm_middle_turn_offset $micron}

The column or row of the HBM channel can be identified in two ways. First, user can specify two bounding boxes, at the two ends of the channel. Bumps inside of the two boxes are routed as part of the channel. The keyword for channel bounding boxes is hbm_bump_array_box, followed by the corner coordinates of the two boxes. If users want to exclude any bumps inside either box, keyword hbm_excluded_bump_in_box must be use for each of the excluded bumps.

{hbm_bump_array_box {llx_box1 lly_box1} {urx_box1 ury_box1} {llx_box2 lly_box2} {urx_box2 ury_box2}} {hbm_excluded_bump_in_box $bump_name}

The second way to describe a channel is to use the inclusion section to specify all the nets that connect two bumps. User also needs to describes any addition bumps in the channel that are not connected to any nets using the keyword hbm_float_bump. Furthermore, if any bump of the channel are connected to components other than front-side micro-bumps, the corresponding net needs to be described using the keyword hbm_2pin_through_net. Both hbm_float_bump and hbm_2pin_through_net are in the exception section.

{hbm_float_bump $bump_name} {hbm_2pin_through_net $net_name}

Two metal layers must be specified for HBM routing. The keywords are hbm_right_array_routing_layer, hbm_left_array_routing_layer, hbm_upper_array_routing_layer and hbm_lower_array_routing_layer. The keywords are followed by the corresponding layer names. The bump arrays at the two ends of the channel are divided into two regions. The bumps in the top or right region are routed using the layer specified with hbm_upper_array_routing_layer or

hbm_right_array_routing_layer. The other bumps in the bottom or left region are routed using the layer specified with hbm_lower_array_routing_layer or hbm_left_array_routing_layer.

{hbm_lower_array_routing_layer $layer_name} {hbm_upper_array_routing_layer $layer_name} {hbm_left_array_routing_layer $layer_name} {hbm_right_array_routing_layer $layer_name}

The routes created by the tool are called stubs. They are used in conjunction with a guidance file during the subsequent RDL routing step. There are two kinds of stubs. The boundary stubs are placed at the boundaries between the bump array regions and the middle routing region of the channel. The routing stubs are inside of the bump array region. The length of stubs can be specified by the keyword hbm_boundary_stublength and hbm_routing_stublength, respectively. The locations of the boundary stubs can be controlled by the keyword hbm_boundary_stub_corners. Specifically, if user choose the keyword auto_align after hbm_boundary_stub_corners, the boundary stub bounding box will be align to the corresponding bump array box. The offsets along the channel direction for the upper and lower end of the channel, high_offset and low_offset, can be given optionally. If not given, the offset values are derived by the tool based on bump dimension and wire routing pitch. Alternatively, user can specify the exact locations for the stubs. The location can be described using the absolute coordinates or the relative offset values with respect to the bump array box as follows.

{hbm_boundary_stub_corners auto_align [high_offset low_offset]} {hbm_boundary_stub_corners absolute | relative x_high y_high x_low y_low}

The boundary stubs are placed on two routing layers. The stubs on the two layers can overlap or be placed in an interleaved pattern. The keyword that controls the stub pattern is hbm_stub_alignment_type. User can choose three pattern as given below. The stub on the lower metal layer is used as the reference.

{hbm_stub_alignment_type inline | staggered_before | staggered_after}

When the keyword **hbm_via_bump_abutment** is false, the locations of boundary stubs can be set to any desired values although the same patterns must be used for both routing layers. Specifically, the gaps between pairs of adjacent boundary stubs are set using keyword hbm_boundary_stub_gaps followed by a list of values in microns. User must specify at least 23 values. Values more than 23 are ignored.

{hbm_boundary_stub_gaps gap_1 gap_2 ... gap_23}

If a repeating pattern exist, the following concise format can be used, in which the repeating pattern is specified in a pair of curly brackets, with the repeating number before.

{hbm_boundary_stub_gaps repeat_times {gap_1 gap_2 ... } }

Although the boundary stubs of high metal layer must be the same as that of the lower metal, the stub locations can be shifted by an offset w.r.t those of the lower metal layer. The offset of is given as below using keyword hbm_boundary_stub_staggered_offset. The value is perpendicular to the channel direction. It can be either positive or negative.

{hbm_boundary_stub_staggered_offset $value}

The keyword **hbm_bump_index_pair** is used in combination with keyword **hbm_boundary_stub_gaps**. User uses **hbm_bump_index_pair** to specify which two nets will be routed next to each other. The format is given below.

```
{hbm_bump_index_pair
{{rowIndex_0_0 column_index_0_0} {rowIndex_0_1 columnIndex_0_1}}
{{rowIndex_1_0 column_index_1_0} {rowIndex_1_1 columnIndex_1_1}}
...
{{rowIndex_N_0 column_index_N_0} {rowIndex_N_1 columnIndex_N_1}}
}
```

The orientation of bump rows is defined as the orientation perpendicular to the channel orientation. The first row and the first column are with the least coordinate value. If the bump is missing at the location specified by the row and column indexes, the bump pair is ignored. If the bump at the location specified by the row and column indexes is not connected to any net, the bump pair is also ignored.

When the keyword **hbm_via_bump_abutment** is true, the Manhattan double-Z style is chosen for the routing in the middle

section of the HBM channel. User can add shielding to the the middle section routing by adding the following in the pattern file.

{hbm_mid_shielding $NET_NAME}

The shielding will have the same width as the corresponding signal routes. The shielding wires are logically assigned to the net $NET_NAME. It is the user's responsibility to keep the routing width less than the routing spacing so that that routing result is valid. In an HBM routing procedure, two layers of metal are used for signal routing. The metal layer beneath either layer is potentially used for the shielding. If the shielding metal cannot be derived, the tool will error out. If both signal routing layers are used, both shielding layers are used. In case that only one signal routing layer is occupied, due to missing nets, only the shielding layer between the two signal routing layers is used for shielding. The shielding routes are not placed above nor below the signal routes. Instead the shielding routes are aligned to the gaps between the signal routes. In this routing style, the signal routings are evenly spaced. Therefore, a shielding route can potential be shared by the signal routes on both sides.

If a channel contains less than 48 nets, the channel is called incomplete channel. An incomplete channel has the same number of bumps but a smaller number of nets. In case that the number of nets is very small, user can force the routing to occur in portions of the bump array regions using the keyword as hbm_boundary_stub_range follows.

{hbm_boundary_stub_range $range_low_upper $range_high_upper $range_low_lower $range_high_lower}

The values are in microns. The pair ($range_low_upper $range_high_upper) and ($range_low_lower $range_high_lower) define corridors in the upper and lower bump array regions, respectively where routing shapes are placed. Since the range is reduced, user must ensure the range is wide enough for all existing nets. When **hbm_boundary_stub_range** and **hbm_via_bump_abutment** are used together, the tool performs a more agressive routing style to ensure routeability than the case without **hbm_boundary_stub_range**. Vias can be abutted to any edge of bumps.

A stub is placed along the routing track, connecting to corresponding bump access via if the following is specified in the pattern file. The stubs are only on the high metal layer.

{hbm_bump_access_dir true}

The length of the stubs can be controlled by the following keyword hbm_bump_access_length. The default value is half routing width plus routing spacing.

{hbm_bump_access_length $DISTANCE}

The tool created two routing guide files for the subsequent RDL routing step. User can provide a prefix for the file names using the keyword route_guidance_file. The actual file names are created by combining the prefix with the corresponding metal layer names. The default prefix is hbm_p2p.txt.

{route_guidance_file $prefix}

User can use the keyword **hbm_stacked_via_dimension** to specify the configuration of any low-level vias placed beneath the interposer vias connected to bump. This keyword is used in the **exception** as follows.

{hbm_stacked_via_dimension $x_size $y_size}

All low-level vias share the same configuration. The default values are 2x2 when **hbm_via_bump_abutment** is false. When the keyword **hbm_via_bump_abutment** is true, the default values are 2x6 and 6x2 for vertical channels and horizontal channels, respectively.

A new keyword **hbm_middle_layer_switch** is introduced in the **exception** section in the **single_hbm** style. When it is used, a special routing pattern is used for the HBM channel routing. Specifically, two metal layers are used to route each net between corresponding bump-access vias. This routing style is only applicable when **hbm_via_bump_abutment** is false. The location of switching is in the middle of two bumps but not overlaps with any routing direction switch in the middle of the HBM channel.

During HBM routing, a detour is used to route around a via. The default detour spacing is **pitch-via_dimension/2-wire_width/2**. A keyword **hbm_detour_spacing** is added so that users can use it in the **exception** section to specify the exact value of detour spacing, namely the distance between the via and detour route. All routing detours share the same value. Users are responsible for the correctness of the value.

{hbm_detour_spacing $micron}

For **single_hbm** routing style, the two bumps of the same net should be at the same relative position in their corresponding bump arrays. In practice, bump assignment mismatch could occur. By default, the tool will error out after detecting bump assignment mismatch and ask users to change bump-net assignment. However, if the assignment cannot be changed, user can use keyword **hbm_stub_order_fix** in the **exception** section to force the tool to proceed. The tool will try to route with a high tolerant to wirelength mismatch. The tool will error out if no solution can be derived.

If the **vertical_routing** style is chosen, the keyword **vertical_route** can be used in the **exception** section to describe how the route mesh will be created. An example is given below. The first string following the keyword describes the metal layer. It is followed by the width and spacing of the routes to be created on that layer. The default width and spacing is the dimension of a single-size via beneath the layer. The last integer is optional. It specifies how many tracks to be created. If the track count is not provided, tracks will be added to cover the entire C4. The dimension of vias between routes are derived based on the track widths.

{vertical_route $layer_name $micron $micron $integer}

DESCRIPTION

This command places vias and routes for an interposer based on the netlist. It has four modes of operation depending on the style of interposer: default, single_hbm, vertical_routing and fpga. In the default mode, the command will only insert vias. In the single_hbm or vertical_routing or fpga mode, both vias and routes can be added. You can provide a pattern file which contains constraints for via and route insertion. The vias inserted by this command can be a single via (base via) or via array which connects a front side bump on the redistribution layer to a metal layer below. Any other vias that are stacked below the base via can also be created based on the specification in the pattern file. Any blockages and preroutes on metal layers related to new vias are considered so that no design-rule violations will occur. If any bumps or nets already have vias placed, no redundant vias will be added. After vias are added, you can use any router to complete the routing of the interposer.

The single_hbm mode is for a single column or row of a high-bandwidth memory (HBM) channel. The tool creates vias and routes according to constraints in the pattern file. It also creates routing guidance files which are used by the subsequent RDL routing command to complete the routing. Many keywords for HBM channel route planning are used in the exception section of the pattern file. They are not applicable to other modes.

The vertical_routing mode is for the connection between a C4 bump and one or two micro-bumps. Given a high-speed net with a C4 bump and one or two micro-bumps, the tool will add vias and routing shapes to connect the C4 to the micro-bump or micro-bumps. Specifically, a RV via is added for each micro-bump. The via is abutted to its micro-bump. The top layer of the via is the layer of micro-bumps. If there is a single micro-bump, the default location of the via is along the top edge of the micro-bump. The via is entirely inside the C4 bump. If the default location of via is not inside the C4 bump, the via will be placed along the bottom edge. If the via cannot be placed inside the C4 bump, the tool will error out. If there are two micro-bumps, the default location is along the top or right edge, when the two micro-bumps are aligned horizontally or vertically, respectively. A single route on the bottom layer of the via is added. If there is a single micro-bump on the net, the route will be horizontal. Otherwise, the orientation of the route is the same as the alignment of the two micro-bumps. On the layers between the top routing layer and layer of C4 bump pins, routing tracks are created to form connection mesh. Specifically, the routings on the layer just below the top routing layer are perpendicular to the routing on the top routing layer. The routing orientation then switches between horizontal and vertical in a round-robin manner in the following layers below until reaching the layer of the C4 pins.

EXAMPLES

The following example places vias by using the default style.

prompt> **create_interposer_routeplan**

SEE ALSO

route_3d_rdl(2)
route_auto(2)
route_eco(2)

create_io_break_cells

Inserts I/O break cells into the floorplan.

SYNTAX

status **create_io_break_cells**
[-reference_cells *lib_cell_name_list*]
[-cells *cell_name_list*]
[-location start | end | both | *offset* | *cell*]
io_guide_list

Data Types

lib_cell_name_list string
cell_name_list string
offset float
cell string
io_guide_list list

ARGUMENTS

-reference_cells *lib_cell_name_list*

Specifies the library cell names of the break cells. For each name in the list, the command inserts a new cell instance with the name __added_break_cell_index, where index is an integer. All new cells are placed consecutively with no gap in between.

-cells *cell_name_list*

Specifies the instance names of the break cells. All cell instances must already exist in the design. Only cell instance names are supported, collections are not supported.

-location start | end | both | *offset* | *cell*

Specifies the location of the break cells to insert with respect to the corresponding I/O guide. You can specify only one guide when using this option.

When the keyword *start* or *end* is specified, the break cells are placed right before or after the I/O guide, respectively. You can also place the break cells at both ends of the I/O guide. If you specify **-location both**, only the **-reference_cells** option can be used.

The *offset* specifies the distance from the starting point of the I/O guide. When a *cell* is specified, the break cells are placed next to the specified cell, away from the guide starting point. If this cell is not placed in the guide, projection is used to determine the break cell location. That is, the cell is projected along its orientation direction toward the I/O guide. The crosspoint will be the location. Note that the cell must be in the correct orientation with respect to the I/O guide. Otherwise, no break cell are inserted.

io_guide_list

Specifies the list of I/O guides. This option is required.

DESCRIPTION

This command places one or more break cells at the intersection of two I/O guides or a specified location in one I/O guide. When one guide is given, the break cell location is determined by the **-location** option.

When two I/O guide names are given, the break cell location is derived from the two guides. If the two I/O guides are perpendicular, the location is the crossing point of the two I/O guides. If the two I/O guides are in parallel, they must have the same orientation. In addition, the extended lines of the two guides must overlap. There must be a gap between the two I/O guides. Otherwise, the command will error out. The break cell location always starts from the gap endpoint of the first I/O guide and extends toward the second I/O guide.

The orientations of the break cells are determined by the first I/O guide and the orientations of their reference cells. Specifically, if the first I/O guide has side set to bottom, the break cell orientations are the same as those of the reference cells. Namely, the break cell instances will be in R0 orientation. If the first guide is with different side values, the break cells will be rotated in the same way as that of IO guide with respect to a bottom guide. User can change the reference cell orientation by setting the reference_orientation attribute of the reference cell. Consequently, break cell orientations will be changed accordingly.

If the break cells are already in the netlist, use the **-cells** option. Otherwise, use the **-reference_cells** option. You can specify only one of the *-cells* and **-reference_cells** options.

EXAMPLES

The following example inserts a break cell with the instance name "break_left" between the io guides io_guide_left and io_guide_top.

```
prompt> create_io_break_cells -cells break_left {io_guide_left io_guide_top}
```

The following example inserts a break cell with the reference name "LEFT_BREAK".

```
prompt> create_io_break_cells -reference_cells LEFT_BREAK {io_guide_left io_guide_top}
```

The following example inserts a break cell with the reference name "LEFT_BREAK" after a pad cell instance named "PAD".

```
prompt> create_io_break_cells -reference_cells LEFT_BREAK -location PAD io_guide_left
```

SEE ALSO

create_io_filler_cells(2)
place_io(2)

create_io_corner_cell

Inserts an I/O corner cell into the floorplan.

SYNTAX

status **create_io_corner_cell**
[-reference_cell *lib_cell_name*]
[-cell *cell_name*]
[-force *block_pathname*]
io_guide_list

Data Types

lib_cell_name string
cell_name string
io_guide_list list
block_pathname string

ARGUMENTS

-reference_cell *lib_cell_name*

Specifies the library cell name of the corner cell. Only one name can be given. The command inserts a new cell instance with the name __added_corner_cell_*index*, where *index* is an integer. This option is mutually exclusive with the **-cell** option.

-cell *cell_name*

Specifies the instance name of the corner cell to be placed. Only one instance name can be given. This option is mutually exclusive with the **-reference_cell** option.

io_guide_list

Specifies the name of the two I/O guide names between which to place the corner cell. This option is required. The two guides must be perpendicular to each other.

-force *block_pathname*

Specifies the parent cell of the newly created corner cell. This option only takes effect if the newly created corner cell using -reference_cell overlaps with multiple hierarchical blocks. The bounding box of the new cell must overlap with the specified block path name.

DESCRIPTION

This command places a corner cell at the intersection of two I/O guides. The two I/O guides must be perpendicular. The location is the crossing point of the two I/O guides.

The orientation of the corner cell is determined by the first I/O guide and the orientation of its reference cell. Specifically, if the first guide is an IO guide with side set to bottom, the corner cell's orientation is the same as that of the reference cell. If first guide is with different side values, the corner cell be rotated in the same way as that of IO guide with respect to a bottom guide. User can change the reference cell orientation by setting the reference_orientation attribute of the reference cell. Consequently, corner cell orientation will be changed accordingly. For example, if users set the reference_orientation to R270, the corner cell will be in R0 orientation when the first guide has its side set to left.

If the corner cell is already in the netlist, use the **-cell** option. Otherwise, use the **-reference_cell** option. You can specify only one of the *-cell* and **-reference_cell** options.

EXAMPLES

The following example inserts a corner cell with the instance name "lt_corner" between the I/O guides io_guide_left and io_guide_top.

```
prompt> create_io_corner_cell -cell lt_corner {io_guide_left io_guide_top}
```

"SEE ALSO"

create_io_filler_cells(2)
place_io(2)

create_io_filler_cells

Inserts I/O filler cells into the specified I/O guides.

SYNTAX

status **create_io_filler_cells**
[-io_guides *io_guide_list*]
[-reference_cells *cell_list*]
[-overlap_cells *cell_list*]
[-extension_bbox *bounding_box*]
[-prefix *prefixString*]
[-top]
[-block *list_of_block_path_names*]
[-force *block_path_name*]
[-auto_tie_to_pg]

Data Types

io_guide_list collection
cell_list collection
bounding_box list
prefixString string
list_of_block_path_names string
block_path_name string

ARGUMENTS

-io_guides *io_guide_list*

Specifies the names or collection of the target I/O guides. If this option is not used and the **-top** and **-block** options are not used, all I/O guides which are logically on top are selected.

-reference_cells *cell_list*

Specifies the lib cell names of the filler cells. These filler cells cannot overlap regular signal or power pad drivers. The reference cells must have difference widths. If there are two filler cells with the same width, only one is used.

-overlap_cells *cell_list*

Specifies the lib cell names of the filler cells that can overlap with regular signal or power pad drivers. The reference cells must have difference widths. If there are two filler cells with the same width, only one is used.

-extension_bbox *bounding_box*

Specifies the bounding box in which filler cells are placed. The filler cells can only be placed in the specified guides and the

bounding box.

-prefix *prefixString*

Specifies the added filler cell name prefix with *prefixString*. If this option is not specified, the command will use the default prefix string __added_filler_instance.

-top

Specifies the tool to select all I/O guides which are logically on top.

-block *list_of_block_path_names*

Specifies the tool to select I/O guides from the mentioned block path names.

-force *block_path_name*

Specifies the parent cell of the newly created filler cell. This option only takes effect if the newly created filler cell overlaps with multiple hierarchical blocks. The bounding box of the new cell must overlap with the specified block path name.

-auto_tie_to_pg

Specifies the tool to connect newly inserted filler cells to logical nets based on physical abutment.

DESCRIPTION

This command inserts filler cells into I/O guides. New filler cells are named __added_filler_instance_*index*, where *index* is a positive integer.

The orientation of a filler cell is determined by the orientation of its reference cell and the I/O guide which the filler cell is placed in. Specifically, if the filler cell is placed in an I/O guide with side set to bottom, its orientation is the same as that of the reference cell. If the filler cell is placed in I/O guides with different side values, the filler cell will be rotated in the same way as that of I/O guide with respect to a bottom guide. You can change the reference cell orientation by setting the reference_orientation attribute of the reference cell. Consequently, the filler cell orientations will be changed accordingly. For example, if you set the reference_orientation to MY, a filler cell placed in a bottom guide will be flipped on the y-axis.

In case, the user specifies the -auto_tie_to_pg option all the terminals of all pins of the newly inserted filler cells for a gap are collected as well as the terminals of the prexisting cells in that gap. The net connection is made for the newly inserted filler cells based on physical contact between the pins. A transitive connection relationship is applied to establish the net connection. The nets are only connected to the filler pins if no errors are reported during filler placement.

The **-block** and **-top** options are mutually exclusive with the **-io_guides** option.

EXAMPLES

The following example inserts filler cell "FILLER" into all I/O guides.

prompt> **create_io_filler_cells -reference_cells FILLER**

SEE ALSO

create_io_break_cells(2)
create_io_corner_cell(2)
create_io_guide(2)
get_io_guides(2)
place_io(2)
remove_io_guides(2)

create_io_guide

Creates an I/O guide in the current design.

SYNTAX

collection **create_io_guide**
[-name *io_guide_name*]
-line { {*x y*} *length* }
-side left | right | bottom | top
[-offset {*start_gap end_gap*}]
[-pad_cells *pad_cell_list*]
[-min_pitch *min_pitch*]

Data Types

io_guide_name string
x float
y float
length float
start_gap float
end_gap float
pad_cell_list list
min_pitch float

ARGUMENTS

-name *io_guide_name*

Specifies the name of the I/O guide. If this option is not used, the I/O guide is named default_guide_*index*, where *index* is a positive integer.

-line { {*x y*} *length* }

Creates an I/O guide along the horizontal or vertical line specified by a starting point {*x y*} and a length *length*. The orientation of the guide is defined by the **-side** option. Left and right lines must be vertical, and top and bottom lines must be horizontal. This is a required option. The line defines the outer edge of a target placement area for the pads associated with the I/O guide. The coordinates are relative to the chip origin.

-side left | **right** | **bottom** | **top**

Indicates the orientation of the I/O guide by specifying the outer edge. For instance, the I/O pads of a left guide are aligned on the right side of the guide line. All guides follow a clockwise orientation. For example, the starting point of a left guide is the bottom end of the line, and the starting point of a right guide is the top point of the line. This option is a required option.

-offset {*start_gap end_gap*}

Specifies the minimal distance (*start_gap*) between the first pad and the starting point of the guide and the minimal distance (*end_gap*) between the last pad and the ending point of the guide. By default, both offsets are zero.

-pad_cells *pad_cell_list*

Specifies a list of pad cells to be included in the I/O guide. The list can contain names, patterns, or collections. It is an error to include a pad cell whose reference module is not a pad type or a cell that has already been assigned to another I/O guide. This option is not required, you can create an empty I/O guide without specifying a pad cell list.

-min_pitch *min_pitch*

Specifies a minimum pitch between I/O drivers on the I/O guide. The distance between two consecutive I/O drivers can be zero, one or multiple times of the min pitch value. Min pitch will be honored during place_io.

There are some limitations. a. If there are conflicts between min pitch and signal I/O constraints or power I/O constraints, the tool will honor signal and power I/O constraints first. b. If there are multiple rings in design, only when there's no overlap between pads and I/O guides can the result be guaranteed.

DESCRIPTION

The **create_io_guide** command creates a line-based I/O pad constraints for I/O placement. The I/O placer uses the I/O guide to properly place the pads within the region. Unlike a move bound or group bound, I/O guides are ordered. Ordering is based on the constraints set by the **set_signal_io_constraints** command.

Note that it is possible to create an inconsistent I/O guide. There might be insufficient space within the specified line to fit the I/O pad cells with the required edge and spacing between pads. Secondly, there is nothing to enforce the position of the pad cells relative to the I/O guide. It is expected that the I/O placer will ensure that the I/O guide constraints are honored. If no *pad_cell_list* is provided as an argument, the command creates an empty I/O guide. You can add pad cells to the I/O guide with the **add_to_io_guide** command and remove pad cells from the I/O guide with the **remove_from_io_guide** command.

EXAMPLES

The following example creates an I/O guide named "io_guide1". The guide starts at location {100 100}, and extends north by 100 due to the **-side left** argument. The guide contains the pads "pad1" and "pad2", which align to the right of the I/O guide line.

```
prompt> create_io_guide -name "io_guide1" -line {{100 100} 100} \
 -side left -pad_cells { pad1 pad2 }
```

SEE ALSO

add_to_io_guide(2)
create_io_ring(2)
get_io_guides(2)
remove_from_io_guide(2)
remove_io_guides(2)
report_io_guides(2)

set_signal_io_constraints(2)

create_io_ring

Creates an IO ring in the current design.

SYNTAX

collection **create_io_ring**
[-name *ring_name*]
[-inside *outer_ring_name*]
[-offset *offset*]
[-guides *io_guide_list*]
[-bbox *bounding_box*]
[-corner_height *height*]
[-pad_cell_list *pad_list*]

Data Types

ring_name string
outer_ring_name string
offset float
io_guide_list list
bounding_box bBox
height float
pad_list list

ARGUMENTS

-name *ring_name*

Specifies the string used as the ring name. If the command creates new IO guides, the ring name is used as the prefix for the IO guides of the ring. Specifically, the left, right, top, and bottom IO guides are named by adding .left, .top, right, and .bottom to the ring name, respectively. By default, the ring name is named _default_io_ring*index*, where index is a positive integer.

-inside *outer_ring_name*

Specifies the name of the (outer) IO ring that is used as the outer boundary for the new ring. When this option is used, the target bounding box is the inner perimeter of the specified outer ring, if the outer ring has a valid corner height. Otherwise, the target bounding box is the bounding box of the outer ring.

-offset *offset*

Indicates the required offset from the target boundary. Namely, the bounding box of the ring is computed as a rectangle inside of the target bounding box. The value specified with the **-offset** option is the distance between the target bounding box and the outer bounding box of the newly created ring. The default offset is 0.

-guides *io_guide_list*

Specifies a list of IO guides to be included in the IO ring. The list may contain names, patterns, or collections. This command does not perform any checking, you are responsible for the correctness of the IO guides and the guides should define a valid ring structure. If you specify an IO guide that is already assigned to another IO ring, the command issues an error message. This option cannot be used with any other options except for the **-name** option. When you use this option to create an IO ring, the corner height of the ring remains at 0.

-bbox ***bounding_box***

Specifies the bounding box for the ring. If this option is not used and the target bounding box of the ring cannot be determined with other options such as **-inside**, the default bounding box is the inner bounding box of the smallest IO ring, or the die area if no IO ring exists.

-corner_height ***height***

Specifies the distance between the outer bounding box of the ring and inner bounding box of the ring. When you use the **-guides** option to create an IO ring, the corner height of the ring remains 0 until you set it to a different value. The default value is the height of the tallest pad cell instances assigned to the ring, or the height of the shortest pad cell instances in the design if no pad cell instance is assigned to the ring.

-pad_cell_list ***pad_list***

Lists the pad cell instances to assign to IO guides inside the ring. The pads will be evenly assigned to the IO guides if the total width of pads in each IO guide is not more than the guide length. This list is not required and you can create a IO ring without specifying a pad cell list.

DESCRIPTION

This command creates or groups a related set of IO guides that are used for IO placement. Note that it is possible to create an inconsistent IO ring, especially for the inner rings of a concentric set. If the **-guides** option is not used, the command creates a default IO ring. The size of the IO ring is assigned using the following priority:

- The *-bbox* argument
- The inner bounding box of the outer ring, if given
- The boundary for the design
- The inner boundary of the smallest existing ring

The command also creates four IO guides: _default_io_ring1.left, _default_io_ring1.bottom, _default_io_ring1.right, and _default_io_ring1.top. You can add additional IO guides to the IO ring with the **add_to_io_ring** command. You can also remove IO guides from the IO ring with the **remove_from_io_ring** command. You can also assign pad cell instances to the ring during the creation of the ring or add pad instances to each guide in the ring at a later time.

EXAMPLES

The following example creates two IO rings and populates them with pad cells:

```
prompt> create_io_ring -name io_ring1
prompt> add_to_io_guide io_ring1.bottom {ioPad_1 ioPad_2 ioPad_3 ioPad_4}
prompt> add_to_io_guide io_ring1.left {ioPad_5 ioPad_6 ioPad_7 ioPad_8}
prompt> add_to_io_guide io_ring1.top {ioPad_12 ioPad_11 ioPad_10 ioPad_9}
```

```
prompt> add_to_io_guide io_ring1.right {ioPad_16 ioPad_15 ioPad_14 ioPad_13}
prompt> create_io_ring -name io_ring2 -inside io_ring1
prompt> add_to_io_guide io_ring2.bottom {ioPad_17 ioPad_18 ioPad_19}
prompt> add_to_io_guide io_ring2.left {ioPad_20 ioPad_21 ioPad_22}
prompt> add_to_io_guide io_ring2.top {ioPad_25 ioPad_24 ioPad_23}
prompt> add_to_io_guide io_ring2.right {ioPad_28 ioPad_27 ioPad_26}
```

The following example creates four IO guides (io_guide1, io_guide2, io_guide3, and io_guide4), then places the guides into an IO ring named "io_ring1".

```
prompt> create_io_guide -name io_guide1 -side left -line {{0 0} 2000}
prompt> create_io_guide -name io_guide2 -side top -line {{0 2000} 2000}
prompt> create_io_guide -name io_guide3 -side right -line {{2000 2000} 2000}
prompt> create_io_guide -name io_guide4 -side bottom -line {{2000 0} 2000}
prompt> create_io_ring -name "io_ring1" -guides { io_guide1 io_guide2 io_guide3 io_guide4 }
```

SEE ALSO

add_to_io_guide(2)
add_to_io_ring(2)
create_io_guide(2)
get_io_guides(2)
get_io_rings(2)
remove_from_io_guide(2)
remove_from_io_ring(2)
remove_io_guides(2)
remove_io_rings(2)
report_io_guides(2)
report_io_rings(2)

create_keepout_margin

Creates design and cell keepout margins.

SYNTAX

collection **create_keepout_margin**
[-type hard | soft | hard_macro | routing_blockage]
[-outer *margin_spec*]
[-inner *margin_spec*]
[-layers *layers*]
[-tracks_per_macro_pin *number_of_tracks*]
[-min_padding_per_macro *padding_size*]
[-max_padding_per_macro *padding_size*]
[*design_and_cell_list*]

Data Types

margin_spec list
layers list
number_of_tracks float
padding_size float
design_and_cell_list list

ARGUMENTS

-type hard | soft | hard_macro | routing_blockage

Specifies the type of the keepout margin to create. Valid values are

- **hard** (the default)
  A hard keepout margin prevents objects from being placed in the keepout margin area throughout the entire flow. However, it would not prevent hard macros being placed in the keepout margin area.
- **soft**
  A soft keepout margin prevents the coarse placer from placing objects in the keepout margin area, but optimization and legalization can place objects within the keepout margin. Also, it would not prevent hard macros being placed in the keepout margin area.
- **hard_macro**
  A hard macro keepout margin prevents hard macros from being placed in the keepout margin area throughout the entire flow.
- **routing_blockage**
  A routing blockage keepout margin prevents PG routes from being placed in the keepout margin area.

-outer *margin_spec*

Specifies the padding margin to maintain outside of the design or cell boundary. This setting is used for physical designs and cells.

Use the following syntax to specify the margins:

{*left bottom right top*}

The margin parameters must be zero or positive and all four parameters are required. The directions are relative to the core area. The units are microns.

-inner *margin_spec*

Specifies the padding margin to maintain inside the design boundary. This option is not valid for leaf cells.

Use the following syntax to specify the margins:

{*left bottom right top*}

The margin parameters must be zero or positive and all four parameters are required. The directions are relative to the core area. The units are microns.

-layers *layers*

Specifies the layers for the routing_blockage type keepout. This option must by specified with, and can only be used with the **-type routing_blockage** keepout. Providing an empty layer list causes an error. Specify the layers by using the layer names from the technology file.

-tracks_per_macro_pin *number_of_tracks*

Specifies the number of tracks per macro pin. This option setting is used to calculate the keepout margin based on wire tracks. When you specify **-tracks_per_macro_pin**, the outer keepout margin is calculated by multiplying the number of tracks by the number of pins. For example, if you set *number_of_tracks* to 0.5, and the macro contains 100 pins on the side, the keepout margin for that side is 50 tracks.

100 pins * 0.5 tracks/pin = 50 tracks

The **-tracks_per_macro_pin** is mutually exclusive with the **-inner** and **-outer** options.

-min_padding_per_macro *padding_size*

Specifies the minimum padding distance per macro. This value is used to calculate the keepout margin based on pin count, and is valid only with the **-tracks_per_macro_pin** option. By default, the minimum padding is the minimum grid size, which is 0.

-max_padding_per_macro *padding_size*

Specifies the maximum padding size per macro. This value is used to calculate the keepout margin based on pin count, and is valid only with the **-tracks_per_macro_pin** option. By default, there is no maximum padding value.

design_and_cell_list

Specifies the list of physical designs and cells for which to create keepout margins.

DESCRIPTION

The **create_keepout_margin** command creates keepout margins for the specified physical designs and cells.

The outer and inner keepout margins can be set explicitly with the **-outer** and **-inner** options. Outer keepouts are allowed only on leaf cells and physical designs. Inner keepouts are allowed only on physical designs and are not supported on leaf cells.

Outer keepout margins can be created for macro cells based on the pin count by using the **-tracks_per_macro_pin** option. The minimum and maximum margins can be specified using the **-min_padding_per_macro** and **-max_padding_per_macro** options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates the soft outer keepout margin on a leaf cell.

```
prompt> create_keepout_margin -type soft \
  -outer {10 10 10 10}  [get_cells MY_LEAF_CELL]
```

The following example creates the hard outer and inner keepout margins on a design.

```
prompt> create_keepout_margin -type hard \
  -outer {10 10 10 10} -inner {5 5 5 5} [get_designs MY_DESIGN]
```

The following example creates the hard outer and inner keepout margins on a physical design.

```
prompt> create_keepout_margin -type hard \
  -outer {10 10 10 10} -inner {5 5 5 5} [get_cells MY_BLOCK]
```

The following example creates the hard outer keepout margin based on the pin count of the macro.

```
prompt> create_keepout_margin -tracks_per_macro_pin .6 \
  -min_padding_per_macro .1 -max_padding_per_macro 0.2 [get_cells MY_MACRO]
```

The following example creates the routing_blockage keepout margin for the PG router on a macro cell for the specified layers.

```
prompt> create_keepout_margin type routing_blockage \
  -outer {10 10 10 10} -layers {M1 M3 M5} [get_cells MY_MACRO]
```

SEE ALSO

get_attribute(2)
get_keepout_margins(2)
remove_keepout_margins(2)
report_keepout_margins(2)

create_layer

Creates a layer in the given tech object.

SYNTAX

collection **create_layer**
[-tech *tech_object*]
-name *layer_name*
-number *layer_number*
[-purpose *purpose_object*]
[-layer_type *layer_type*]
[-mask_name *mask_name*]
[-before *reference_layer*]
[-after *reference_layer*]

Data Types

layer_name	string
layer_number	int
layer_type	string
mask_name	string
reference_layer	string or collection

ARGUMENTS

-tech

Layer will be created in this tech object. If not specified then tech object of current library will be used.

-name *layer_name*

Specifies the name of the layer to be created.

-number *layer_number*

Specifies the number of the layer to be created.

-purpose

Layer will be created using this purpose object. If not specified then the default purpose used for layer creation is "drawing". This option is mutually exclusive with -before and -after.

-layer_type *layer_type*

Specifies the type of the layer to be created. Valid values are: **diffusion**, **implant**, **interconnect**, **local_interconnect**, **local_via_cut**, **mim_interconnect**, **mim_via_cut**, **n_implant**, **nwell**, **p_implant**, **place_and_route**, **pwell**, **sadp_trim**, **unknown**,

and **via_cut**. Default layer type is unknown.

-mask_name *mask_name*

Specifies the mask_name of the layer to be created.

-before *reference_layer*

Specifies the mask_order of the layer to be created relative to the given reference layer. The reference layer should be an ordered layer with valid mask_order before which the created layer will be inserted. Please note that this option would increment the mask_order of all higher order layers including the given reference layer to create space in the mask_order sequence for insertion of the created new layer. If unspecified, the mask_order of the created layer is -1 (unordered). This option is mutually exclusive with -after and -purpose.

-after *reference_layer*

Specifies the mask_order of the layer to be created relative to the given reference layer. The reference layer should be an ordered layer with valid mask_order after which the created layer will be inserted. Please note that this option would increment the mask_order of all higher order layers excluding the given reference layer to create space in the mask_order sequence for insertion of the created new layer. If unspecified, the mask_order of the created layer is -1 (unordered). This option is mutually exclusive with -before and -purpose.

DESCRIPTION

create_layer command creates a layer in the given technology. The layer creation requires a name and a number for the new layer. If a purpose is specified, then the new layer uses the given purpose, otherwise the new layer uses the default purpose "drawing".

If the command is successful, it returns the newly-created layer as a single-element collection. If the command fails, it returns TCL_ERROR with appropriate error message. If there is a syntax error, it returns a TCL_ERROR.

EXAMPLE

The following example creates a layer in the technology of the current design.

prompt> **create_layer -tech [get_techs [current_lib]] -name test_layer -number 100 -purpose signal**
{test_layer:signal}

The following example creates the same layer without -tech option.

prompt> **create_layer -name test_layer -number 100 -purpose signal**
{test_layer:signal}

The following example creates the same layer with mask_name and mask_order after layer M1.

prompt> **create_layer -name test_layer -mask_name test_layer -after M1 -number 100**
{test_layer}

SEE ALSO

remove_tech(2)
get_techs(2)

create_left_right_filler_cells

Inserts left or right filler cells for specific center cells.

SYNTAX

status **create_left_right_filler_cells**
-lib_cells *center_left_right_pair_list*
[-prefix *prefix*]
[-boundaries *polyrect_list*]
[-voltage_areas *va_list*]
[-rules *rule_list*]
[-follow_stdcell_orientation]

Data Types

center_left_right_pair_list	list
prefix	string
polyrect_list	list
va_list	list
rule_list	list

ARGUMENTS

-lib_cells *center_left_right_pair_list*

Specifies the list of center cell list, left filler cell list and right filler cell list pair. The format should be {{{center_cell_list1} {left_filler_list1} {right_filler_list1}} {{center_cell_list2} {left_filler_list2} {right_filler_list2}}...}. By default, the tool adds the left right filler cells in the order that you specify. We recommend to specify them from the largest to smallest. The center cell list, left filler list and right filler list allow wildcard "*" specification. If wildcard is used for left filler list and right filler list, the order will be the same with that you get from **get_lib_cells**. The center cell list can't be empty. Either the left or right filler list can be empty. For example, {{center_cell_list} {left_filler_list} {}} or {{center_cell_list} {} {right_filler_list}}. But left and right filler lists can't be empty simultaneously. Besides, a lib cell can't be repeated in two or more center cell lists. If so, only the last specification will be used.

-prefix *prefix*

Adds the specified prefix to the instance names of left right filler cells inserted by this command. You can later use the prefix to identify the filler cell instances.

-boundaries *polyrect_list*

Specifies the rectangular or rectilinear polygon regions (in microns) where left right filler cells are to be inserted. A rectangle is specified as a list of two points, {{*llx lly*} {*urx ury*}}, which represent the lower-left and upper-right corners of the rectangle.

By default, left right filler cells are inserted in the entire chip.

-voltage_areas *va_list*

Specifies the voltage areas in which to insert left right filler cells.

By default, the tool inserts left right filler cells in all voltage areas.

-rules *rule_list*

Specifies the special rules to follow during left right filler cell insertion.

You can specify one or more of the following keywords:

- **no_1x**
  Indicates this is no-1x design. Prevents tool from placing left right filler cells such that it would leave a 1x gap.

-follow_stdcell_orientation

Specifies the left and right filler cells to follow the center cell's orientation. If the center cell's orientation is R180 or MY, its left and right side will be swapped. Besides, its left right filler cells' orientation will be MY on R0 rows and R180 on MX rows. If the center cell's orientation is R0 or MX, its left right filler cells' orientation will be R0 on R0 rows and MX on MX rows.

By default, the tool will place the left right filler cells with the allowable orientation of the corresponding rows.

DESCRIPTION

This command inserts left or right filler cells for the specified center cells. The height of center cells must be multiple integer of the height of the left or right filler cells. There will be no gaps between the left or right filler cells and its corresponding center cells. Intercell spacing rules will be ignored.

After the left or right filler cells insertion, you may use check_legality to verify filler placement legality.

EXAMPLES

The following example inserts left and right filler cells for center cells whose reference library named mylib/Center_1 and it also inserts left filler cells for center cells whose reference library named mylib/Center_2. All the left and right filler cells will only be inserted within the specified region {{0 0} {500 500}}. If the sizes of the left filler cells are such that LeftFiller_4X > LeftFiller_2X > LeftFiller_1X, this ordering minimizes the number of left filler cells added by using the larger filler cells first. And it was the same case for the right filler cells.

```
prompt> create_left_right_filler_cells \
  -lib_cells { { {mylib/Center_1} \
               {mylib/LeftFiller_4X mylib/LeftFiller_2X} \
               {mylib/RightFiller_2X mylib/RightFiller_1X} } \
             { {mylib/Center_2} \
               {mylib/LeftFiller_2X mylib/LeftFiller_1X} \
               {} } \
           } \
  -boundaries {{0 0} {500 500}}
```

SEE ALSO

check_legality(2)

create_length_limit

Creates an analog constraint group with wire_length_limit intent in the current design.

SYNTAX

collection **create_length_limit**
[*constraint_group_name*]
-for *objects*
[-min_value *float*]
[-exclude *objects*]
[-driver *objects*]
[-force]

Data Types

constraint_group_name string
objects collection
float float

ARGUMENTS

constraint_group_name

Specifies the name of the constraint group. If not specified then a system generated name is assigned.

-for *objects*

Specifies the objects - can be nets, pins, ports, bundles, or topology_edges - that constitute the analog constraint group. It can be specified as a string (name of objects) or the collection of objects.

-min_value *float*

Specifies the minimum value that needs to be achieved by each constituent object in the constraint group. The default is zero.

-exclude *objects*

Specifies the objects that need to be excluded from the analog constraint group. The objects can be pins or ports, and can be specified as strings (object names) or collections. This option is useful when user gives a net which is connecting physically (across logical hierarchies) to multiple pins / ports in **-for** option but want to exclude a few pins / ports connecting to that net.

-driver *objects*

Specifies the object to be treated as the driver for the analog constraint group. The object can be a pin or port, and can be specified as a string (object name) or collection. This option is useful when user gives a net which is connecting physically (across logical hierarchies) to multiple pins / ports in **-for** option but want to control the length of each driver to receiver connection.

-force

Specifies force deletion of existing constraint group by the same name as specified and re-creation of new constraint group. The default is false and the command exits with error if same name is encountered.

DESCRIPTION

The **create_length_limit** command creates a constraint group for a collection of nets, pins, ports, bundles, and topology_edges with wire_length_limit intent. Each constituent object must honor the minimum value specified for its wire length. Constraint applicable on pin/port check the net which is connecting pin/port to driver pin/port. The constraint is used by Custom Router.

EXAMPLES

The following example creates a constraint group named 'abc' for nets and bundles matching the pattern 'pr*' such that none of the constituent objects have wire length lesser than 10. The 'pr*' nets are routed with Custom Router.

```
prompt> create_length_limit abc -for pr* -min_value 10
{abc}
```

SEE ALSO

get_constraint_groups(2)
remove_constraint_groups(2)
report_constraint_groups(2)

create_lib

Creates a design library.

SYNTAX

collection **create_lib**
[-technology *tech_path* | -use_technology_lib *tech_lib_name*]
[-scale_factor *scale_factor*]
[-ref_libs *ref_libs*]
[-convert_sites *site_name_pairs_list*]
[-base_lib *base_library_path*]
library_name

Data Types

tech_path string
tech_lib_name string
scale_factor int
ref_libs list
site_name_pairs_list list
base_library_path string
library_name string

ARGUMENTS

-technology *tech_path*

Specifies the technology file for this library.

-use_technology_lib *tech_lib_name*

Specifies the reference library to use as a dedicated technology library.

The library must contain a technology section and must be one of the libraries specified with the **-ref_libs** option.

This option is mutually exclusive with the **-technology** option, and it must be used with the **-ref_libs** option.

-scale_factor *scale_factor*

Specifies the length precision for this library. The length precision for the library and all of its reference libraries must be identical. The value is specified in terms of units per micron. By default, a length precision of 10000 is used, which implies one internal unit is equal to one Angstrom or 0.1 nm

Choose a *scale_factor* value that will result in a whole number of database units per minimum grid spacing. For example, if your minimum grid spacing is 1 nm, the default scale_factor of 10000 dbu per micron could be used. But if your minimum grid spacing is 0.25 nm, a scale factor of, for example, 4000, must not be used because rounding errors might result.

-ref_libs ***ref_libs***

Specifies the reference libraries for this library.

In addition to pre-built cell libraries, this option accepts physical source data such as physical libraries (which contain frame views of the cells), LEF files, and Milkyway libraries. The tool automatically builds cell libraries based on the physical source data.

You can specify the reference libraries and physical source data using absolute or relative paths. If you specify the libraries with a relative path or with no path, the tool uses the search path defined with the **search_path** variable to locate the libraries.

-convert_sites ***site_name_pairs_list***

Specifies the mapping for the site names in the technology file in the following format:

{*tech_file_site_name new_site_name*}

This option must be used with the **-technology** option.

-base_lib ***base_library_path***

Creates a sparse library based on the specified library. The base library must be a design lib and cannot be a lib-cell library. No other options can be specified when the **-base_lib** option is used. For example, the sparse library must use the same technology file and ref_libs as the base library.

library_name

Specifies the name of the new library. This can be a simple, relative, or absolute path. If it is a relative or absolute path, the trailing portion of the path after he last '/' becomes the name of the library. It is an error if a library, file, or directory with the same name already exists on disk.

The name can be a simple path, a path relative to the current working directory, or an absolute path. The name cannot contain spaces or the colon (":") character, which is used to delimit the library and design name.

The *library_name* is optional when *-base_lib* option is used. If unspecified, the name of the created local sparse library matches the name of the base library specified using the *-base_lib*.

DESCRIPTION

The **create_lib** command creates a new design library. The library can contain technology information specified with the **-technology** option. If there are sites defined in the technology file (and one is being used), you can use the **-convert_sites** option to map the sites from one name to another. This same option is used with the **read_def** command to ensure that the site names match among all the input data.

You can directly specify the library's reference library path list with the **-ref_libs** option. Alternatively, if you specify physical source data with the **-ref_libs** option, the tool creates the cell libraries and sets them as reference libraries. The supported physical source data includes physical libraries, which contain the frame views of the cells, LEF files, and Milkyway libraries. The logic libraries are determined from the **link_library** variable. The created cell libraries are put in the directory specified by the **lib.configuration.local_output_dir** application option. You can use the **lib.configuration.default_flow_setup** application option to point to a Tcl script with customized settings for the cell library creation. See the man pages of these application variables and options for detailed information about how to use them.

When the **-base_lib** option is used the **create_lib** command creates a new sparse library based upon an existing design library on disk. The sparse library uses the same technology file and ref_libs as the base library. Initially, a sparse library does not have any locally stored blocks. It copies the base library's catalog and ref_lib list into memory as its own and also remembers the full path of the base library for reference.

The library is then opened and made the current library.

If the command is successful, it returns the newly-created library as a single-element collection. If the command fails, it returns an empty string. If there is a syntax error, it returns a TCL_ERROR.

EXAMPLES

The following example creates a library named r4000 in the current directory that uses the tcbn90g.tf technology file.

```
prompt> create_lib -technology LIBS/TECH/tcbn90g.tf r4000
Information: Loading technology file '/usr/LIBS/TECH/tcbn90g.tf' (FILE-007)
{r4000}
prompt> current_lib
{r4000}
```

The following example creates a library named r4000 in the current directory that uses the tcbn90g.tf technology file and has a scale factor of 4000.

```
prompt> create_lib -technology LIBS/TECH/tcbn90g.tf r4000 -scale_factor 4000
Information: Loading technology file '/usr/LIBS/TECH/tcbn90g.tf' (FILE-007)
{r4000}
prompt> current_lib
{r4000}
prompt> get_attribute [current_lib] scale_factor
4000
```

The following example creates a library named top_lib in the ./lib_dir directory that uses the tcbn90g.tf technology file.

```
prompt> create_lib -technology LIBS/TECH/tcbn90g.tf lib_dir/top_lib
Information: Loading technology file '/usr/LIBS/TECH/tcbn90g.tf' (FILE-007)
{top_lib}
```

The following example creates a library named design_lib in the /home/user directory that uses the my_tech.tf technology file.

```
prompt> create_lib -technology my_tech.tf /home/user/design_lib
Information: Loading technology file 'my_tech.tf' (FILE-007)
{design_lib}
```

The following example creates a sparse library named top_sparse_lib based on the design library top_lib in the /home/user directory.

```
prompt> create_lib -base_lib /home/user/top_lib top_sparse_lib
Information: Creating Sparse View library 'top_sparse_lib' with base
library '/home/user/top_lib'. (NDM-103)
{top_sparse_lib}
```

The following example creates a library named design_lib and automatically creates cell libraries for it. The cell libraries are created based on the logic library files specified in the **link_library** variable and the physical library specified with the **-ref_libs** option.

```
prompt> set_app_var link_library \
  "* 1_pvt1.db 1_pvt2.db 2_pvt1.db 2_pvt2.db"
prompt> create_lib -technology my_tech.tf /home/user/design_lib \
  -ref_libs frame_only.ndm
 ... run lm_shell to build reference libraries.
 ..........................................................................
Information: Successfully built 3 reference libraries: 1.ndm 2.ndm
EXPLORE_physical_only.ndm
```

{design_lib}

SEE ALSO

open_lib(2)
save_lib(2)
close_lib(2)
copy_lib(2)
move_lib(2)
current_lib(2)
get_libs(2)
set_ref_libs(2)
set_base_lib(2)
search_path(3)
link_library(3)
lib.configuration.local_output_dir(3)
lib.configuration.default_flow_setup(3)

create_liner_cells

Create liner cells along die edge or core edge in the block.

SYNTAX

create_liner_cells
-lib_cell *cell_name*
[-horizontal_padding_cells *cells_name*]
[-vertical_padding_cells *cells_name*]
[-die_offset *{x_offset y_offset}*]
[-macro_offset *{x_offset y_offset}*]
[-core_offset *{x_offset y_offset}*]
[-step *{x_step y_step}*]
[-along_core_edge]

Data Types

cell_name string
cells_name list
x_offset float
y_offset float
x_step float
y_step float

ARGUMENTS

-lib_cell

Specifies the library liner cell to be inserted.

-horizontal_padding_cells

Specifies a list of padding cells to replace liner cell and fill up the horizontal gaps.

-vertical_padding_cells

Specifies a list of padding cells to replace liner cell and fill up the vertical gaps.

-die_offset

Specifies the horizontal and vertical enclosure from die boundary to the corner liner cell. The default value is {0 0}.

-macro_offset

Specifies the minimum horizontal and vertical spacing from macro boundary to the corner liner cell. The default value is {0 0}.

-core_offset

Specifies the minimum horizontal and vertical spacing from core edge to the corner liner cell if '-along_core_edge' is not specified. The default maximum value is twice as x_step and y_step. If '-along_core_edge' is specified, it specifies the exact horizontal and vertical spacing from core edge to the corner liner cell. The default value is {0 0}.

-step

Specifies the pitch of two adjacent liner cells. The x_step and y_step must be equal to or greater than liner cell's width and height. The default value is liner cell's width and height.

-along_core_edge

Specifies whether to insert liner cell along core edge. If not specified, liner cells will be only inserted along die edge (in die area). Otherwise, liner cells will be also inserted along core edge (outside core area).

DESCRIPTION

This command creates liner cells by keep specified spacing to die edge, macro and core edge and inserts horizontal and vertical padding cells when necessary.

EXAMPLES

The following example shows creating liner cells along both die edge and core edge with potential horiontal and vertical padding cells.

```
create_liner_cells -lib_cell LC -horizontal_padding_cells {HPC1 HPC2 HPC3} -vertical_padding_cells {VPC1 VPC2} -step {3 4} -di
```

SEE ALSO

create_logic_net

Creates a logic net

SYNTAX

status **create_logic_net**
net_name

Data Types

net_name string

ARGUMENTS

net_name

Specifies the name of the net to create.

DESCRIPTION

The create_logic_net command creates a logic net in the current scope.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

create_net(2)

create_logic_port

Creates a logic port

SYNTAX

string **create_logic_port**
port_name
[-direction in | out | inout]

Data Types

port_name string

ARGUMENTS

port_name

Specifies the name of the port to create.

-direction in | out | inout

Specifies the direction of the port. The default is **in**.

DESCRIPTION

The create_logic_port command creates a logic port in the current scope. Logic ports should be created before corresponding isolation and level-shifting strategies are applied so any isolation or level-shifting strategy defined for a power domain may apply to logic ports created on the boundary of that power domain.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

create_port(2)

create_macro_array

Arranges macros into an array.

SYNTAX

collection **create_macro_array**
-num_rows *rows*
-num_cols *columns*
[-horizontal_channel_height *list_of_numbers*]
[-vertical_channel_width *list_of_numbers*]
[-align *align_mode*]
[-orientation *list_of_orientations*]
[-flip_alternate_rows]
[-flip_alternate_cols]
[-create_group true | false]
[-name *edit_group_name*]
[-fill_pattern by_row | by_col | by_row_snake_pattern | by_col_snake_pattern]
macros

Data Types

rows integer
columns integer
list_of_numbers list
align_mode string
list_of_orientations list
edit_group_name string
macros collection

ARGUMENTS

-num_rows *rows*

Specifies the number of rows in the array.

-num_cols *columns*

Specifies the number of columns in the array.

-horizontal_channel_height *list_of_numbers*

Specifies the height of each horizontal channel. The height for each macro is set based on its position in the list. If there are fewer elements in the list than there are channels, the remaining channel heights are set to the first element of the list. In particular, if only one number is specified, all heights are set to that number. The default for all heights is 0. Note that heights are calculated from macro edges; keepout margins are not considered.

-vertical_channel_width *list_of_numbers*

Specifies the width of each vertical channel. The width for each macro is set based on its position in the list. If there are fewer elements in the list than there are channels, the remaining channel widths are set to the first element of the list. In particular, if only one number is specified, all widths are set to that number. The default is for all widths is 0. Note that the widths are calculated from macro edges; keepout margins are not considered.

-align *align_mode*

Specifies the edge or corner of each macro to which to align for non-homogeneous macro arrays. The valid values are **center**, **top**, **bottom**, **right**, **left**, **top_left**, **bottom_left**, **top_right**, and **bottom_right**.

The default is **center**.

-orientation *list_of_orientations*

Specifies the orientation for each macro in the array, based on its position in the list. If there are fewer elements in the list than there are macros, the remaining macro orientations are set to the first element of the list. In particular, if only one orientation is specified, all macro orientations are set to that value. The default is for all orientations to be unchanged from the current value.

Legal orientation values are: N, FN, S, FS, E, FE, W, and FW

- N - specifies an R0 orientation
- FN - specifies an MY orientation
- S - specifies an R180 orientation
- FS - specifies an MX orientation
- E - specifies an R270 orientation
- FE - specifies an MYR90 orientation
- W - specifies an R90 orientation
- FW - specifies an MXR90 orientation

-flip_alternate_rows

Flip macros in every other row. By default, macros are not flipped.

-flip_alternate_cols

Flip macros in every other column. By default, macros are not flipped.

-create_group true | false

Specifies whether to create an edit group for the macro array. The default is true.

-name *edit_group_name*

Specifies the name of edit group to create. This option is valid only if **-create_group true** is specified. If **-name** is not specified and -create_group is true, a unique default name, MACRO_ARRAY_#, is used to name the macro. If **-name** is specified and the specified name already exists in another edit group, the command issues an error message.

-fill_pattern by_row | by_col | by_row_snake_pattern | by_col_snake_pattern

Specifies how macros are placed in the array. by_row places macros starting from the first column; new rows are started from the first column. by_row_snake_pattern creates the array row-by-row, but with each row starting where the previous row ended, thus creating a snake pattern.

by_col and by_col_snake_pattern are similar, but place macros column-by-column rather than row-by-row The default fill pattern

is by_row.

macros

Specifies a list of macros to place in the array.

DESCRIPTION

This command arranges a list of macros in an array and optionally creates an edit group for the array. If an edit group is created, the command returns it; otherwise a collection equal to the list_of_objects argument is returned.

EXAMPLES

The following example creates an array from the selected macros.

```
prompt> create_macro_array -num_rows 2 -num_cols 2 [get_selection] -name my_edit_group
```

The following example creates an array from the selected macros and specifies their channels widths, and heights. The orientation for each macro in the array is N.

```
prompt> create_macro_array -num_rows 4 \
  -num_cols 3 [get_selection] -orientation { N } \
  -horizontal_channel_height 10 \
  -vertical_channel_width { 10 20 }
```

The following example creates an array from the selected macros and specifies their orientations, channels widths, and heights.

```
prompt> create_macro_array -num_rows 4 \
  -num_cols 3 [get_selection] -orientation { N S N S } \
  -horizontal_channel_height 10 \
  -vertical_channel_width { 10 20 }
```

The following example returns all macro arrays (edit groups that begin with "MACRO_ARRAY_") and deletes them.

```
prompt> set macro_arrays [get_edit_groups -filter name=~MACRO_ARRAY_*]
{MACRO_ARRAY_0}
prompt> remove_edit_groups $macro_arrays
1
```

SEE ALSO

create_edit_group(2)
get_edit_groups(2)
remove_edit_groups(2)

create_macro_groups

Create macro groups.

SYNTAX

collection_of_macro_group **create_macro_groups**
-method auto | by_hierarchy | by_connectivity | by_pattern | by_index_pattern | by_macros
[-names string]
[-pattern pattern_string]
[-index_pattern index_pattern_string]
[-macros list_of_macros]
[-preview]

Data Types

pattern_string Regular expression
index_pattern_string Index pattern expression
list_of_macros collectionP of macros

ARGUMENTS

-method auto | by_hierarchy | by_connectivity | by_pattern | by_index_pattern | by_macros

Specifies method to create macro group. There are five modes to create macro groups. They are auto, by_hierarchy, by_connectivity, by_pattern, by_index_pattern, and by_macros.

auto is the default method. This method is the same as default grouping method of **create_placement -floorplan** command. It groups macros based on combinations of hierarchy, connectivity, and coarse placement result.

by_hierarchy is the same grouping by hiererachy method of **create_placement -floorplan** command. It groups macros purely based on hierarchy infomration.

by_connectivity is the same grouping by connectivity of **create_placement -floorplan** command. It groups macros based on conneectivity between macros and sequential cells.

by_pattern method groups macros based on name pattern. Parameter -pattern must be specified to use this method. Pattern is a regular expression string. Macros which match the pattern and belong to same placeable area will be grouped together. For example, assume there are four macros

HIER1/MEM_1_1
HIER1/MEM_1_2
HIER1/MEM_1_3
HIER1/MEM_1_4

and they belong to same block.

```
prompt> create_macro_groups -method by_pattern \
   -pattern HIER1/MEM_1*
```

This command will create a macro group which contains these four macros. If one macro belongs to another macro group before this command, the command will remove it from its previous macro group and assign it to the new macro group.

by_index_pattern method group macros based on index pattern. Parameter -index_pattern must be specified to use this method. Index pattern is comprised of fixed text and index. Index has two types, fixed and auto. Fixed index is represent as ?, fixed index is represet as *. Fixed index is used to group macros of same placeable area together. For example, assuming there are following macros

```
HIER/MEM_1_1
HIER/MEM_1_2
HIER/MEM_2_1
HIER/MEM_2_2
```

and they belong to same placeable area.

```
prompt> create_macro_groups -method by_index_pattern \
  -index_pattern HIER/MEM_?_*
```

This command spcifies index pattern as HIER/MEM_?_*. Question mark (?) is fixed index, and star (*) is auto index. The command will first find all macros which match the pattern, i.e, macros with name HIER_MEM_number1_number2. Since the index pattern indicates the first number is fixed, and second is auto, the command will group macros with same first number together. So, in the end, the command will create two groups:

```
HIER/MEM_1_1
HIER/MEM_1_2

and

HIER/MEM_2_1
HIER/MEM_2_2
```

Following command uses a different index_pattern.

```
prompt> create_macro_groups -method by_index_pattern \
   -index_pattern HIER/MEM_*_?
```

The index pattern indicates the first number is auto, and the second number is fixed. The command will create two groups and they are

```
HIER/MEM_1_1
HIER/MEM_2_1

and

HIER/MEM_1_2
HIER/MEM_2_2
```

by_macros method creates macro group based on given macros. Paramenter -macros must be specified to use this method. The command will create a macro group which contains macros in the list.

-names

Specifies group names. The command **create_macro_groups** can create multiple groups. This parametre specify names for each group. It is a list of string. Each item is a group name. If the number of names specified in this parameter is more than macro groups created by the command, additional names will be discarded. If the number of names specified in this parametre is less than macro groups created by the command, the command will create default names for additional macro groups. If any name is already used by other macro group, the command will give a warning message and pick a default name.

-pattern regular_expression_pattern

Specifies regular expression name pattern of macros. This parameter must be used together with parametre -method set as by_pattern.

-index_pattern index_pattern_string

Specifies index pattern of macros. This parameter must be used together with parametre -method set as by_index_pattern.

-preview

With -preview parameter, the command **create_macro_groups** will return a list of collections, and each collection is a macro group which will be created if parameter -preview is not specified.

DESCRIPTION

This command creates macro groups. It is one of command of macro placement assistant flow. Macro placement assistant flow helps user to group and pack macros semi-automatically. This is different from command **create_placement -floorplan**, where the grouping and packing of macros are done automatically.A typical macro placement assistant flow is as following:

\n+[step]. Place macros. Macros can be placed by i**create_placement -floorplan**
command automatically, or by user manually.
\n+[step]. Use **create_macro_groups** command to create macro groups. User
can use different methods to create macro groups. Macros of same macro
groups should be placed together.
\n+[step]. Create contour of macro groups using command set_macro_group_shape.
\n+[step]. Pack macros using command pack_macro_group.

User can undo the change at any step if he or she is not satisfied with the result. in NDM model, macro group is a sub-class of movebound. It contains a group of macros and contour. All macros of one macro group must belong to the same macro group. A placeable area can be a movebound, a voltgage area, or a block. When a macro group is first created, its contour will be placed outside of block area. Use command set_macro_group_shape to modify the contour.

EXAMPLES

The following command creates macro groups with auto method

prompt> **create_macro_groups** -method auto

The following ocmmand creates macro group with give macros

prompt> **create_macro_groups** -method by_macros \
-macros [get_cells {M1 M2 M3 M4}]

The following command print macro groups which will be created by method by_pattern.

prompt> **create_macro_groups** -method by_pattern \

```
    -pattern  I_ORCA_TOP/I_CONTEXT_MEM/I_CONTEXT_RAM_* \
    -preview
```

SEE ALSO

add_array_to_macro_group(2)
remove_array_from_macro_group(2)
add_macro_to_group(2)
remove_macros_from_group(2)
clone_macro_group_packing(2)
get_macro_group_packing_clone_candidates(2)
remove_macro_groups(2)
split_macro_group(2)
pack_macro_group(2)
unpack_macro_group(2)
set_macro_group_shape(2)

create_macro_relative_location_placement

Performs macro placement based on the relative location placement constraints.

SYNTAX

status **create_macro_relative_location_placement**
[-hierarchical]

ARGUMENTS

-hierarchical

Performs macro relative location placement in the top-level design and in the child blocks. By default, the command performs macro relative location placement in the current block.

DESCRIPTION

This command performs macro relative location placement based on the constraints for current block. If the **-hierarchical** option is specified, the command places the macros in the current block and the macros in child blocks.

EXAMPLES

The following example performs relative location placement on current block.

```
prompt> create_macro_relative_location_placement
1
```

The following example performs relative location placement on the top-level block and in the child blocks.

```
prompt> create_macro_relative_location_placement -hierarchical
1
```

SEE ALSO

create_placement(2)
remove_macro_relative_location(2)
report_macro_relative_location(2)
set_macro_relative_location(2)

create_marker_layers

Populates a set of shape objects in the marker layers in the given design.

SYNTAX

collection **create_marker_layers**
[-design *design*]
[-cells *cells*]
[-references *references*]
[-horizontal_extension *hextension*]
[-vertical_extension *vextension*]
[-horizontal_offset_from_cell *hoffset*]
[-vertical_offset_from_cell *voffset*]
[-exclude { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects]

Data Types

design collection
cells collection
references string
hextension string
vextension string
hoffset string
voffset string
llx float
lly float
urx float
ury float
x float
y float
geometric_objects collection

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-cells *cells*

Specifies the collection of cells which are to be excluded for generating the shapes in the marker layers.

-references *references*

Specifies the names of the IPS which are to be excluded for generating the shapes in the marker layers.

-horizontal_extension *hextension*

Specifies the horizontal extension for marker layers where the tracks are vertical. The input is a list of layername and extension pairs. The extension value can be positive or negative.

-vertical_extension *vextension*

Specifies the vertical extension for marker layers where the tracks are horizontal. The input is a list of layername and extension pairs. The extension value can be positive or negative.

-horizontal_offset_from_cell *hoffset*

Specifies the horizontal offset from excluded cells or references for generating the shapes of the marker layers.The input is a list of layername and offset pairs. The offset value can be positive or negative.

-vertical_offset_from_cell *voffset*

Specifies the vertical offset from excluded cells or references for generating the shapes of the marker layers.The input is a list of layername and offset pairs. The offset value can be positive or negative.

-exclude { {{*llx lly*} {*urx ury*}} | {{*x y*} {*x y*} {*x y*} {*x y*}...} } | geometric_objects

Specifies the objects to be excluded from the marker layers. The object list can be a heterogeneous collection of poly_rect, geo_mask, shape, layer, and other physical objects.

DESCRIPTION

This command creates a collection of new rectangles and polygons on the marker layers and returns a collection that contains the new shape.

EXAMPLES

The following example creates shapes on the marker layers.

```
prompt> create_marker_layers -references {IP1 IP3} -cells u1 \
  -horizontal_extension {{{M2_P48 10} {M3_P48 20}}}
```

The following example creates shapes on the marker layers where offset from cells and references are specified.

```
prompt> create_marker_layers -references {IP1 IP3} -cells u1 \
  -horizontal_extension {{{M2_P48 10} {M3_P48 20}}}
  -horizontal_offset_from_cell {{{M2_P48 6} {M3_P48 8}}}
```

SEE ALSO

get_shapes(2)
remove_shapes(2)

create_mask_constraint_routing_blockages

Creates routing blockages around the terminals and boundaries in the current top-level design, or around the block pins and inside the block instance boundaries on the layers that have mask constraints defined.

SYNTAX

status **create_mask_constraint_routing_blockages**
[-cells *cells*]
[-create_placement_blockages]
[-self]

Data Types

cells collection

ARGUMENTS

-cells *cells*

Specifies a list of block instances in which to create the mask constraint routing blockages. If this option is omitted, the command creates mask constraint routing blockages inside each of the block instances within the current top-level design.

The command places mask constraint routing blockages inside the specified block instances (or all of the block instances if omitted). The blockages surround the block pins on the layers where mask constraints are defined.

-create_placement_blockages

Creates placement blockages in addition to routing blockages. The shapes and locations of the placement blockages are the union of the mask constraint routing blockages.

-self

Specifies that the mask constraint routing blockages are created at top-level design. The mask constraint routing blockages are created in the top-level design and surround the top-level terminals on layers where mask constraints are defined. This option is mutually exclusive with the **-cells** option.

When this option is omitted, mask constraints routing blockages are created inside the block instances within the current top-level design.

DESCRIPTION

This command creates mask constraint routing blockages, and optionally creates placement blockages. If the technology file for the

current top-level design defines mask constraint rules on one or more layers, this command creates routing blockages. If the **-self** option is specified, routing blockages are created around the terminals and boundaries of the top-level design. If the **-self** option is not specified, the command creates routing blockages around the block pins and inside the block instances specified by the **-cells** option. If both **-self** and **-cells** options are omitted, the command creates routing blockages inside all of block instances. Blockages are created on the layers that have the mask constraint rules defined. The **set_editability** can enable/disable this command for blocks or levels of physical hierarchy. This command has no effect on the blocks that are disabled by **set_editability**.

Routing blockages prevent the router from accessing those terminals or the block pins on the same layers within a certain distance of the boundaries and avoid creating mask constraint violations.

The routing blockages created by this command use the naming style __DPT_ROUTING_BLOCKAGE_x__, where x is a nonnegative integer. If the design already contains a routing blockage with the same naming style, the tool deletes the blockage automatically.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates double-patterning routing blockages for the current top-level design.

```
prompt> create_mask_constraint_routing_blockages
```

SEE ALSO

create_routing_blockage(2)
set_editability(2)

create_matching_type

Creates a matching type in the current block.

SYNTAX

collection **create_matching_type**
-name *matching_type_name*
[-uniquify *count*]
[*object_list*]

Data Types

matching_type_name string
count int
object_list list

ARGUMENTS

-name *matching_type_name*

Specifies the name of the matching type. The name must be unique among all matching types in the block.

-uniquify *count*

Specifies how bump cells are assigned to pad pins within the same matching type. This is optional. If specified, the valid value range is [-4, 8]. If the value is > 0, the value indicates the number of pad pins that can be assigned to a single bump cell, and no two bump cells can be assigned to the same pad pin. If the value is < 0, the absolute value indicates the number of bump cells that must be assigned to each pad pin, and no two pad pins can be associated to the same bump cell. If the value = 0, only 1 bump cell is assigned to all pad pins.

object_list

Specifies a list of cells, pins, and terminals to add to the matching type. The list can contain names, patterns, or collections. This list is optional. You can create an empty matching type without specifying any object list. It is an error to add an object which is already in another matching type. Each object can exist in at most one matching type in the block.

DESCRIPTION

This command defines a matching type constraint. A matching type specifies which bump cells are to be connected to which pad pins through its list of objects. It also optionally specifies how this is done through its uniquify number. If the matching type contains only bumps or no bumps, no bump assignment will be performed on its list of objects.

When I/O placement is performed, objects can be connected only to other objects with the same matching type. The matching type exists only in the block in which it was created, but its containing objects can span into subblocks. This means that the current block's matching types can apply to objects in subblocks when performing I/O placement from the current block, but matching types created in other blocks are not visible from the current block.

A pin is a RDL pin if it contains at least a single terminal with class attribute equal to bump. When a pad is in a matching type, all its RDL pins are in the matching type. When a RDL pin with multiple terminals with class attribute equal to bump is included in a matching type, only a single terminal will be selected for bump assignment and RDL routing. To force the tool to connect all, users need to create a matching type with all terminals explicitly described. Bumps and the uniquify number must also be set accordingly.

EXAMPLES

The following example creates a matching type named "myMatchingType1" with cell "bump" and pin "padCell/padio".

```
prompt> create_matching_type -name "myMatchingType1" -uniquify 0 \
  [list [get_cells bump] [get_pins padCell/padio]
```

SEE ALSO

add_to_matching_type(2)
get_matching_types(2)
place_io(2)
remove_from_matching_type(2)
remove_matching_types(2)
report_matching_types(2)

create_mim_capacitor_array

Places Metal-Insulator-Metal (MIM) capacitor cells in the design in an array style. When inserting MIM capacitor cells, the command ignores placement blockages, macros, standard cells and PG network routes.

SYNTAX

status **create_mim_capacitor_array**
-lib_cell *lib_cell*
-x_increment *distance*
-y_increment *distance*
[-prefix *prefix*]
[-orientation R0|R90|R180|R270|MX|MY|MXR90|MYR90]
[-boundary *polygon*]

Data Types

lib_cell	string
distance	float
prefix	string
polygon	list

ARGUMENTS

-lib_cell *lib_cell*

Specifies the MIM capacitor library cell explicitly. This option is required.

-x_increment *distance*

Specifies the horizontal distance between the centers of two cells.

-y_increment *distance*

Specifies the vertical distance between centers of two cells.

-prefix *prefix*

Specifies the prefix added to the MIM capacitor cell name.

-orientation R0|R90|R180|R270|MX|MY|MXR90|MYR90

Specifies the orientation of the MIM capacitor cells placed by this command. The orientation can be R0, R90, R180, R270, MX, MY, MXR90, MYR90. If not specified, the default orientation is R0.

-boundary *polygon*

Specifies the area in which to place MIM capacitor cells.

DESCRIPTION

This command places the cells into the design in an array style.

EXAMPLES

The following example places MIM capacitor cells.

```
prompt> create_mim_capacitor_array -lib_cell my_lib/mim_ref_cell \
  -x_increment 20 -y_increment 20
```

SEE ALSO

create_cell(2)

create_mismatch_config

Creates a new config.

SYNTAX

status **create_mismatch_config**
config_name
[-ref_config *name*]

Data Types

config_name string
name string

ARGUMENTS

config_name

Specifies the config name to be created.

-ref_config *name*

Specifies the reference config to be used for creating new config. If no ref_config is passed, then the most restrictive config will be used, i.e. **default**. Pre-defined configs available are: auto_fix and default. One config among pre-defined configs or user defined configs can be used as a reference config.

DESCRIPTION

This command creates a new config. It uses the ref_config to create the new config if passed any. If no ref_config is passed, most restictive config which is **default** is used to create the new config. Pre-defined configs are: default and auto_fix. auto_fix being permissive and default being restrictive.

If the new named config already exists, or reference config does not exists, proper messaging is given. The **report_mismatch_configs** can be used to list all the available configs in a session. So, once **create_mismatch_config** succeeds for a config, **report_mismatch_configs** will report this config too.

The **set_current_mismatch_config** command can be used to set the newly created config as current config of a session.

EXAMPLES

The following example creates new config:

```
prompt> create_mismatch_config user-def
1
prompt> create_mismatch_config user-def
Error: Mismatch config 'user-def' already exists. (DMM-016)
0
```

The following example creates a new config based on a ref config:

```
prompt> create_mismatch_config user-def1 -ref_config auto_fix
1
```

The following example creates a new config using a user defined config:

```
prompt> create_mismatch_config user-def
1
prompt> create_mismatch_config user-def1 -ref_config user-def
1
prompt> create_mismatch_config user-def3 -ref_config user-def10
Error: Mismatch config 'user-def10' do not exists. (DMM-015)
0
```

SEE ALSO

report_design_mismatch(2)
get_mismatch_types(2)
get_mismatch_objects(2)
report_mismatch_configs(2)
set_current_mismatch_config(2)
get_current_mismatch_config(2)

create_mode

Creates a mode in the current design.

SYNTAX

string **create_mode**
mode_name

Data Types

mode_name string

ARGUMENTS

mode_name

Specifies the name of the mode being created.

DESCRIPTION

Creates a mode in the current design. Modes are used to contain a set of constraints environment for the design. Unique modes are often used for different combinations of operating modes and power schemes. Modes are used together with corners, which contain operating condition and parasitics parameters. Timing analysis is actually done for various mode/corner combinations, as configured by the **set_scenario_status** command.

Constraints are contained in the current mode. When the design is linked, a single mode named "default" is automatically created and set to be the current mode. If the **create_mode** command is used, the current_mode is set to the newly-created mode. You can set the current mode to a different mode with the **current_mode** command.

To create a collection of modes matching a pattern and optionally matching filter criteria, use the **get_modes** command. To get a collection of all modes in the design, use the **all_modes** command.

To undo **create_mode**, use the **remove_modes** command.

EXAMPLES

The following example creates a mode named "Mission5" and sets it to be the current mode.

```
prompt> create_mode Mission5
```

SEE ALSO

all_modes(2)
current_mode(2)
get_modes(2)
remove_modes(2)
create_corner(2)
set_scenario_status(2)

create_module

Creates one or more modules within a design.

SYNTAX

collection **create_module**
[-design *design*]
modules

Data Types

design collection
modules list

ARGUMENTS

-design *design*

Specifies the design to create the modules in. If no design is specified, the modules are created in the *current_design*.

modules

Specifies the names of the modules to be created in the design. Each module name must be unique.

DESCRIPTION

This command creates new modules in the current design. A module is a definition of logical hierarchy within a physical design block. The newly created modules are empty, they have no ports or cells. Instances of these modules may then be created with the **create_cell** command.

Unused modules may be deleted with the **remove_modules** command.

EXAMPLES

The following example creates a module named FOO.

```
prompt> create_module FOO
```

The following example creates modules named *MOD1*, *MOD2*, and *MOD3*.

```
prompt> create_module {MOD1 MOD2 MOD3}
```

SEE ALSO

create_cell(2)
current_design(2)
remove_modules(2)

create_multibit

Creates a multibit cell from a list of single-bit cells.

SYNTAX

status **create_multibit**
object_list
[-name *multibit_cell_name*]
-lib_cell *library_name/lib_cell_name*

Data Types

object_list list
multibit_cell_name string
library_name/lib_cell_name string

ARGUMENTS

object_list

Specifies a list of single-bit registers or latches or mv cells in the current design from which a multibit cell is created. The specified cells are removed from the netlist and replaced by the multibit cell. The order in which the cells are listed determines the order in which the nets are connected to the inserted cell.

-name *multibit_cell_name*

Specifies the name of the new multibit cell. If this option is omitted, the command derives a name by concatenating the names of *object_list*.

-lib_cell *library_name/lib_cell_name*

Specifies the name of the library reference cell used for the new cell. The bit-width of the specified library cell must be at least as large as the total bit-width of the cells specified in the *object_list*. You can specify only one library cell name. Specifying the library name along with the cell name (for example, my_lib/my_cell) is mandatory.

DESCRIPTION

The **create_multibit** command creates a new multibit cell from a list of registers or latches or mv cells in the current design. All the single-bit cells in *object_list* are replaced by one multi-bit cell.

The command verifies that the cells in the *object_list* exist in the design and have valid locations, and do not have a "dont_touch" or "fixed" attribute.

The order of the specified cells determines the pin connection order of the new multi-bit cell. For example, a net connected to the third specified cell in the *object_list* will be connected to the third bit of the multibit cell inserted by the command.

If the multi-bit library cell has a larger bit-width than the total bit-width of the specified cells, the pins of the unused bits of the cell are left dangling. If a pin of a specified cell does not have a corresponding pin in the multibit library cell, the tool disconnects the pin and issues a warning message.

To automatically generate a script containing **create_multibit** commands, use the **identify_multibit** command.

EXAMPLES

The following example creates a multibit register cell from 4 single-bit registers:

```
prompt> create_multibit -name group0_0 \
  {reg_1 reg_2 reg_3 reg_4} -lib_cell REG_4_BIT
```

SEE ALSO

identify_multibit(2)
split_multibit(2)

create_multisource_clock_sink_group

Defines sink groups for multisource clock tap assignment

SYNTAX

status **create_multisource_clock_sink_group**
-name *name*
-sinks *pins_or_ports*
[-driver_object *pins_or_ports_or_nets*]
-type exclusive | non_exclusive

Data Types

name	string
pins_or_ports	collection
pins_or_ports_or_nets	collection

ARGUMENTS

-name *name*

Specifies a name for the sink group. This option is required. The name can be used to refer to the sink group for reporting, removal, or modifications. Names have to be unique for all defined sink groups.

-sinks *pins_or_ports*

Specifies a list of clock sinks that should get assigned to the same tap driver as specified with the **-driver_object** option. If the **-driver_object** option is not specified the tool will assign the sinks to the closest tap driver available. A sink can only be assigned once to a sink group. Trying to assign a sink to multiple sink groups results in an error. The clock sinks have to be endpoints of the clock that gets assigned to the driver object using the **set_multisource_clock_tap_options** command.

-driver_object *pins_or_ports_or_nets*

This optional option specifies a driver object to which the sinks shall get assigned during tap assignment using the **synthesize_multisource_clock_taps** command. A driver object can either be a pin or port. If a net name is used, then the driver pin or port of that net is inferred as the driver object. Without using this option the group of sinks is assigned to the closest available tap driver.

-type exclusive | non_exclusive

Specifies the type of the sink group. The type can be either exclusive or non_exclusive. Exclusive means that no other sink which is not in that sink group can be assigned to the tap driver of that sink group. A non-exclusive sink group allows other sinks to get assigned to the tap driver of the sink group as well.

DESCRIPTION

The **create_multisource_clock_sink_group** command is used to define clock sinks to tap drivers. The assignment itself and the required netlist modifications are performed using the **synthesize_multisource_clock_taps** command. The sink group definition will only take effect if the driver object has also been assigned to a clock using the **set_multisource_clock_tap_options** command. Multiple sink groups can be assigned to the same driver object as long as they are non-exclusive.

An error is issued if a particular sink group with the given name has already been defined.

The **remove_multisource_clock_sink_groups** command can be used to remove sink groups that have been created with this command.

The **report_multisource_clock_sink_groups** command can be used to report sink groups created with this command.

The **get_multisource_clock_sink_groups** command can be used to obtain a collection of sink groups.

The **add_to_multisource_clock_sink_group** command can be used to add additional sinks to an existing sink group.

The **remove_from_multisource_clock_sink_group** command can be used to remove sinks from an existing sink group.

Attributes

Four attributes are defined on sink groups:

- name: a read-only attribute that returns the name of a sink group.
- sinks: a read-only attribute that returns the collection of sinks of a sink group.
- type: attribute that corresponds to the type (exclusive or non-exclusive) of a sink group. The attribute is read-write.
- driver_object: The driver of the sink group. The attribute is read-write.

The **get_attribute**, **set_attribute**, and **report_attributes** commands can be used to access these attributes.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This example creates an exclusive sink group of two sinks.

```
create_multisource_clock_sink_group -name sink_group1
 -type exclusive
 -driver_object [get_pins {tap_driver1/Z}]
 -sinks [get_pins {reg1/CK reg2/CK}]
```

This example creates a non-exclusive sink group.

```
prompt> create_multisource_clock_sink_group -name sink_group1 \
  -type non_exclusive \
  -driver_object [get_pins {tap_driver2/Z}] \
  -sinks [get_pins {reg*/CK}]
```

This example creates a non-exclusive sink group without an explicit driver object.

```
prompt> create_multisource_clock_sink_group -name sink_group1 \
  -type non_exclusive \
  -sinks [get_pins {reg*/CK}]
```

SEE ALSO

add_to_multisource_clock_sink_group(2)
get_multisource_clock_sink_groups(2)
remove_from_multisource_clock_sink_group(2)
remove_multisource_clock_sink_groups(2)
report_multisource_clock_sink_groups(2)
set_multisource_clock_tap_options(2)
synthesize_multisource_clock_taps(2)

create_mv_cells

Create power management cells including repeater, isolation and level shifter cells in the design based on UPF strategies.

SYNTAX

status **create_mv_cells**
[-isolation]
[-level_shifter]
[-map_ao_block]
[-mapped]
[-all]
[-generate_strategy level_shifter]
[-strategy_output *output_filename*]
[-all_blocks | -blocks *block_designs*]
[-host_options *host_option_name*]
[-verbose]

Data Types

output_filename string
block_designs collection
host_option_name string

ARGUMENTS

-all

Inserts repeater, isolation and level shifter cells in the design based on repeater, isolation and level shifter strategy settings. When the design has single rail cells without explicit connect_supply_net in AO Block Domain, the tool will remap such cells into dual rail cells.

-all_blocks

Create MV cells for all the child blocks in the current design and the current design itself. This option is mutually exclusive with *-blocks*. Only one of them can be specified at a time.

-blocks *block_designs*

Specifies the list of child block designs to create MV cells. This option is mutually exclusive with *-all_blocks*. Only one of them can be specified at a time.

-host_options *host_option_name*

Specifies the host option to use for distributed processing. If not specified, the tool uses the global host option if the option contains a distributed processing settings.

-isolation

Inserts isolation cells in the design based on isolation strategy settings. If there is no user isolation strategy, isolation cells will not be inserted. The tool automatically triggers repeater insertion based on repeater strategy settings.

-level_shifter

Inserts level shifter cells in the design based on level shifter strategy settings. The tool automatically triggers repeater insertion based on repeater strategy settings.

-map_ao_block

When the design has single rail cells without explicit connect_supply_net in AO Block Domain, the tool will remap the cells into dual rail cells.

-mapped

By default, tool will insert new repeater, level shifter and isolation cells from GTECH library only. When -mapped option is specified, tool will insert new repeater, level shifter and isolation cells from user libraries only. Existing GTECH repeater, level shifter and isolation cells in the design will be updated to user library cells.

-generate_strategy level_shifter

If no level shifter strategy is loaded in the UPF, the tool can generate general level shifter strategies based on domain supply voltage shifting status. If there is no voltage shifting among all domain supplies, no strategy is generated. The **-generate_strategy** option only supports the *level_shifter* keyword.

This option can be combined with the **-strategy_output** option. If **-strategy_output** specifies an output file, strategies generated will be written out to the output file only and the tool will not perform level shifter cell insertion based on the strategy. If **-strategy_output** is not specified, the tool continues to do automatic level shifter insertion based on the strategy generated internally. The strategies can be written out in the UPF derived session.

-strategy_output *output_filename*

If the **-generate_strategy level_shifter** option is specified together with **-strategy_output output_filename**, the tool generates general level shifter strategies based on domain supply voltage shifting status and writes out all strategies to the specified file. You can modify the strategies and reload them into the tool.

-verbose

Prints out additional debug messages when inserting power management cells. Currently the verbose messages only include debugging information for level shifter cell insertion.

DESCRIPTION

This command creates power management cells in the design based on UPF strategies.

In addition, this command applies optimization restrictions (dont-touhces) on the nets between isolation, level shifter, repeaters cells and their corresponding power domain boundaries. Buffers might be inserted on these nets to prevent them from being over-constrained.

EXAMPLES

The following example inserts repeater, isolation and level shifter cells in the design based on UPF strategies:

```
prompt> create_mv_cells -all
Information: Total 20 repeater cells have been inserted. (MV-054)
Information: Total 51 isolation cells have been inserted. (MV-054)
Information: Total 51 isolation cells have been associated. (MV-021)
Information: Total 30 level shifter cells have been inserted. (MV-054)
Information: Total 0 MV restriction buffer cells have been inserted. (MV-054)
1
```

SEE ALSO

check_mv_design(2)
map_isolation_cell(2)
map_level_shifter_cell(2)
set_isolation(2)
set_isolation_control(2)
set_level_shifter(2)

create_net

Creates one or more power, ground, tie, or signal nets.

SYNTAX

collection **create_net**
[-design *design*]
[-power | -ground | -tie_high | -tie_low]
[-cell *cell*]
net_names

Data Types

design collection
cell collection
net_names list

ARGUMENTS

-design *design*

Specifies the design in which to create the nets. If no design is specified, the nets are created in the current design.

-power

Specifies that power nets should be created.

-ground

Specifies that ground nets should be created.

-tie_high

Specifies that tie-high nets should be created.

-tie_low

Specifies that tie-low nets show be created.

net_names

Specifies the names of the nets to be created in the design. Each net name must be unique.

-cell cell

Specifies the cell where the net is to be added. The net is created in the cell's reference module. The cell must not reference a library cell, unless this command is executed in the library manager. In the library manager, nets may be created in library cells.

When not specified, the net is created in the current module.

DESCRIPTION

This command creates new nets in the current design (unless otherwise specified by -design). It creates only scalar (single-bit) nets. One net is created for each of the names listed.

This command can create PG, tie, or signal nets. If no net type is specified, the command creates signal nets.

Nets connect ports and pins in a design. The **create_net** command creates nets, but does not connect the nets. To establish this connection, use the **connect_net** or **connect_pg_net** commands.

EXAMPLES

The following example creates signal nets named *N1*, *N2*, *N3*, and *N4*.

```
prompt> create_net {N1, N2, N3, N4}
```

The following example creates a power net named *VDD*.

```
prompt> create_net -power VDD
```

SEE ALSO

connect_net(2)
disconnect_net(2)
get_nets(2)
remove_nets(2)

create_net_bus

Creates a net bus.

SYNTAX

collection **create_net_bus**
[-design *design*]
[-block *block*]
[-cell *cell*]
[-create_nets]
net_bus_name

Data Types

design collection
block collection
cell collection
net_bus_name string

ARGUMENTS

-design *design*

Specifies the top-level design for finding objects. If this is not specified, objects are found in the current design.

-block *block*

Specifies the block where the net bus is to be added. If specified, **create_net_bus** has the same effect as **edit_module** in which the module is changed and changes are applied to all module occurrences.

-cell *cell*

Specifies the cell where the net bus is to be added. The net bus is created on the cell's reference module. **create_net_bus** first uniquifies the reference if needed, then creates the net bus. When not specified, the net bus is created on the current module.

-create_nets

Creates net bus members if they do not exist.

net_bus_name

Specifies the net bus name and its range using the **name [start: end]** format. Specify a net bus named busA with net from 1 *to* 5 as **busA[1:5]**. Specify a net bus named busB with net from 5 *downto* as **busB [5:1]**.

DESCRIPTION

This command creates a net bus, collects or creates net bus members, and puts them into the net bus created.This supports creating n-dimensional bus.

The command returns the created net_bus(as a collection), an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

EXAMPLES

The following example creates net bus named *bus1* from net 1 to 5 as bus[1], bus[2], bus[3], bus[4], bus[5]

```
prompt> create_net_bus bus1[1:5]
{"bus1"}
```

The following example first creates nets, then creates a net bus named *bus2* from the created net with indexes 5 downto 1 as bus2[5], bus2[4], bus2[3], bus2[2], bus2[1]

```
prompt> create_net_bus bus2[5:1] -create_nets
{"bus2"}
```

SEE ALSO

add_to_net_bus(2)
get_net_buses(2)
remove_from_net_bus(2)
remove_net_buses(2)
report_net_buses(2)

create_net_priority

Creates an analog constraint group with net_priority intent in the current design.

SYNTAX

collection **create_net_priority**
[*constraint_group_name*]
-for *objects*
[-priority *int*]
[-force]

Data Types

constraint_group_name string
objects collection
int integer

ARGUMENTS

constraint_group_name

Specifies the name of the constraint group. If not specified then a system generated name is assigned.

-for *objects*

Specifies the objects - can be nets, bundles, or topology_edges - that constitute the analog constraint group. It can be specified as a string (name of objects) or the collection of objects.

-priority *int*

Specifies the priority of routing for the constituent objects. Objects with higher priority will be routed first. Priority values can be given from -128 to 128. Default is zero.

-force

Specifies force deletion of existing constraint group by the same name as specified and recreation of new constraint group. Default is false where the command exits with error if same name is encountered.

DESCRIPTION

The **create_net_priority** command enables you to create a constraint group for a collection of nets, bundles, and topology_edges with net_priority intent. The constraint is used by Custom Router. All the constituent objects will be routed with same priority and will

be routed before other objects that have lower priority.

EXAMPLES

The following example creates a constraint group named 'abc' for nets and bundles matching the pattern 'pr*' with net_priority intent and priority of 15. The 'pr*' nets are routed with Custom Router.

```
prompt> create_net_priority abc -for pr* -priority 15 {abc}
```

SEE ALSO

get_constraint_groups(2)
remove_constraint_groups(2)
report_constraint_groups(2)

create_net_shielding

Creates an analog constraint group with shielding intent in the current block.

SYNTAX

collection **create_net_shielding**
[*constraint_group_name*]
-for *objects*
[-shield_net *net*]
[-shield_net_2 *net*]
[-gap *distance*]
[-width *distance*]
[-max_gap *distance*]
[-min_segment *distance*]
[-sharing true | false | unset]
[-enclose_pins true | false | unset]
[-enclose_vias true | false | unset]
[-via_defs *via_defs*]
[-disabled_layers *layers*]
[-layer_widths *widths*]
[-layer_gaps *gaps*]
[-layer_max_gaps *gaps*]
[-group_shield]
[-force]

Data Types

constraint_group_name string
objects collection
net collection
distance float
via_defs string
layers string
widths list
gaps list

ARGUMENTS

constraint_group_name

Specifies the name of the constraint group.

By default, the command generates a unique constraint group named shielding_*n*, where *n* is a unique, monotonically increasing integer value.

-for *objects*

Specifies the objects that constitute the analog constraint group. The objects can be nets, bundles, or topology_edges. You can specify the objects as a list of object names or as a collection of objects.

-shield_net *net*

Specifies the net to connect to shield material. For single signal routing, the tool uses this net for the left and bottom shields.

-shield_net_2 *net*

Specifies the net to connect to secondary shield. For single signal routing, the tool uses this net for the top and right shields.

-gap *distance*

Specifies the layer-independent spacing between route objects and shield objects.

The default is 0.

-width *distance*

Specifies the layer-independent width for shielding material.

The default is 0.

-max_gap *distance*

Specifies the layer-independent maximum spacing between route objects and shield objects.

The default is 0.

-min_segment *distance*

Specifies the minimum length of wiring to shield.

The default is 0.

-sharing true | **false** | **unset**

Specifies whether shielding can be shared between constituent objects of the constraint group.

Allowed values are **false**, **true**, and **unset**.

The default is **unset**.

-enclose_pins true | **false** | **unset**

Specifies whether pins are enclosed by the shield. This option is mutually exclusive with *-group_shield* option.

Allowed values are **false**, **true**, and **unset**.

The default is **unset**.

-enclose_vias true | **false** | **unset**

Specifies whether vias are enclosed by the shield.

Allowed values are **false**, **true**, and **unset**.

The default is **unset**.

-via_defs *via_defs*

Specifies the via definition names to be used to connect to the shield.

By default, all via definitions can be used.

-disabled_layers ***layers***

Specifies the layer names that are not to be used for shielding.

-layer_widths ***widths***

Specifies the width of a layer used for shielding.

This is a list of alternating layer and width values, where the layer is a string and the width is a floating point number. For example, {M1 5.0 M2 6.0}.

-layer_gaps ***gaps***

Specifies the gap of shielding material from wiring material.

This is a list of alternating layer and gap values, where the layer is a string and the gap is a floating point number. For example, {M1 5.0 M2 6.0}.

-layer_max_gaps ***gaps***

Specifies the maximum gap of shielding material from wiring material.

This is a list of alternating layer and maximum gap values, where the layer is a string and the maximum gap is a floating point number. For example, {M1 5.0 M2 6.0}.

-group_shield

Specifies whether a single set of shield shapes can be used to shield all nets of the constraint group. This option is mutually exclusive with *-enclose_pins* option.

-force

Forces the deletion of an existing constraint group if the name matches the specified *constraint_group_name* argument.

By default, the command exits with an error if the name already exists.

DESCRIPTION

Some signal nets require isolation or shielding to allow the circuit to function correctly. Whether it is to reduce crosstalk or parasitic effects, some nets must be shielded.

The **create_net_shielding** command creates a constraint group for a collection of nets, bundles, and topology_edges with shielding intent to achieve the necessary shielding. The constraint is used by the Custom Router tool.

Each of the constituent objects are shielded separately as per the requirements specified in the constraint group. Use this constraint to specify which layers to constrain and any layer-specific requirements.

EXAMPLES

The following example creates a constraint group with a system-generated name that describes the shielding intent for nets whose name starts with "pr". A gap of 3 is used for the M2 layer and a gap of 5 is used for the M3 layer. All constituent objects can share the shield and the M5 and M6 layers cannot be used for shielding. The 'pr*' nets are routed with the Custom Router tool.

```
prompt> create_net_shielding -for pr* -disabled_layers {M5 M6} \
  -sharing true -layer_gaps {M2 3 M3 5}
{shielding_4}
```

SEE ALSO

get_constraint_groups(2)
remove_constraint_groups(2)
report_constraint_groups(2)

create_pad_rings

Creates pad rings with simplified options control.

SYNTAX

status **create_pad_rings**
[-create *create_opt*]
[-nets *netname_list*]
-route_pins_on_layer *layer_opt*
[-min_target_layer *layer_opt*]
[-max_target_layer *layer_opt*]
[-drc *drc_opt*]
[-sides *side_opt*]
[-undo]

Data Types

create_opt specification
netname_list list
layer_opt specification
drc_opt specification
side_opt specification

ARGUMENTS

-create all | pg | specified_net

Creates pad rings for all pad pins, all pg pins, or pins with the specified nets. If the **-nets** options is specified, *all* and *pg* will be ignored. This argument is optional.

-nets *netname_list*

Specifies a list of nets on which to create pad rings. This argument is optional.

-route_pins_on_layer *layer_opt*

Creates pad rings on the specified metal layers. This argument is required.

-min_target_layer *layer_opt*

Specifies the lowest metal layer to create via for. This argument is optional.

-max_target_layer *layer_opt*

Specifies the highest metal layer to create via for. This argument is optional.

-drc no_check | check_but_no_fix

Specifies the DRC option for PG pad ring creation. The argument following **-drc** must be either *no_check* or *check_but_no_fix*. *no_check* specifies that no DRC check is performed. *check_but_no_fix* specifies that DRC check is performed and DRC errors are reported, but no DRC fixing is performed. By default, DRC is checked and fixed for the PG pad ring creation.

-sides *side_opt*

Specifies the side option for PG pad ring creation to support connections on aligned pins between neighboring pad cells. The syntax of the *side_opt* argument is *{{side start end} {side start end} ...}* where *side* is one of the following keywords: *left*|*right*|*top*|*bottom*. For *left* and *right* sides, the connection will be vertical and for *top* and *bottom* sides the connection will be horizontal. *start* and *end* denote the coordinate ranges to search for pad cells. It can be inside the core area as well. For *left* and *right* sides they are x coordinates and y for *top* and *bottom* sides.

-undo

Removes pad rings and corresponding vias created by this command and restores the shapes to their original condition before running the **create_pad_rings** command.

DESCRIPTION

This command creates PG pad rings and provides a simple user interface. This command automatically calculates the required information to create pad rings and provides a simpler user interface than the **create_pg_ring_pattern** command.

EXAMPLES

The following example performs PG pad ring creation on metal layer M8 for all PG nets.

```
prompt> create_pad_rings -create pg -route_pins_on_layer {M8}
```

The following command performs PG pad ring creation on metal layer M8 for nets VDD and VSS without checking for DRC violations.

```
prompt> create_pad_rings -nets {VDD VSS} \
   -route_pins_on_layer {M8} -drc no_check
```

The following command connects horizontally aligned pins of adjacent pad cells whose y coordinates fall between 0um and 700 um and between 1000um and 1700um.

```
prompt> create_pad_rings -create all  \
   -route_pins_on_layer {M4 M5} -sides {{bottom 0 700} {top 1000 1700}}
```

SEE ALSO

check_pg_connectivity(2)
compile_pg(2)
create_pg_ring_pattern(2)
get_shapes(2)

get_vias(2)
remove_pg_patterns(2)
report_pg_patterns(2)
report_pg_strategies(2)
set_pg_strategy(2)

create_pg_composite_pattern

Creates a power ground composite pattern. This composite pattern can be defined hierarchically based on low-level patterns such as the basic wire patterns, or other composite patterns.

The composite pattern can be instantiated in a design by the **set_pg_strategy** command to create a power ground mesh.

SYNTAX

status **create_pg_composite_pattern**
pattern_name
[-nets *net_list*]
[-parameters *var_list*]
-add_patterns *pattern_expr*
[-via_rule *via_rule_spec*]

Data Types

pattern_name string
net_list list
var_list list
pattern_expr specification
via_rule_spec specification

ARGUMENTS

pattern_name

Specifies the name of the composite pattern.

-nets *net_list*

Specifies power and ground net names used in this composite pattern. The net names are symbolic names and are mapped from net names specified from a high-level command, that is, either from the symbolic net names in another **create_pg_composite_pattern** command or from the real net names in the **set_pg_strategy** command.

The net mapping is by position. For example, the first net name in a high-level command maps to the first net name in the low-level command, the second net name in a high-level command maps to the second net name in the low-level command, and so on.

If this option is not specified, the list of nets from its high-level command is mapped to its low-level patterns in the same order. If the low-level patterns have different orders or net names, you must specify this option, and specify the order of net names in **-add_patterns** option. If net-based via rules are needed in this composite pattern, net names should also be provided using this option for filtering purpose.

-parameters *var_list*

Specifies the list of parameters. Each parameter can be evaluated and passed into the low-level patterns used by this composite pattern. This allows the pattern to be programmed and reused by specifying different parameters for different situations. Each parameter *var* in the list *var_list* is a string. The "@" character is used as a prefix keyword to evaluate the parameter, for example, @*var*. Refer to the example section for more details.

-add_patterns *pattern_expr*

Specifies the low-level patterns for this composite pattern. Each low-level pattern is either a wire pattern created by the **create_pg_wire_pattern** command, or a composite pattern created by the **create_pg_composite_pattern** command. The low-level pattern cannot be the current composite pattern or contain the current composite pattern as its low-level pattern, that is, you must not create a recursive definition. Each pattern is specified by a pattern expression *pattern_expr* in the following syntax:

{name: *child_pattern_name*}
{nets: *sub_net_list*}
{parameters: *sub_var_list*}
{offset: {*x_offset y_offset*}}
{pitch: {*x_pitch y_pitch*}}
{repeat {*x_num y_num*}}

The **name** keyword followed by *child_pattern_name* specifies the low-level pattern name to add to the current composite pattern.

The **nets** keyword followed by *sub_net_list* specifies the net name list to map into the symbolic list of nets in *child_pattern_name*. Each net name in *sub_net_list* must be one of the net names specified in **-nets** in the current composite pattern. The index of the *i_th* net in *sub_net_list* is *i*. If not specified, the list of nets from the current composite pattern is used for mapping.

The **parameters** keyword followed by *sub_var_list* specifies the parameter list to pass into the low-level pattern *child_pattern_name*. Each parameter can be either a value, such as 0.5 for wire width, or a variable evaluation, such as "@w" for wire width. "w" should be a parameter specified in **-parameters** in the current composite pattern.

The **offset** keyword followed by {*x_offset y_offset*} specifies the horizontal and vertical offsets when applying the low-level pattern *child_pattern_name* in the current composite pattern. The offset is applied to the origin of the low-level pattern *child_pattern_name*. The offset value is relative to the current composite pattern origin. By default, the offset is 0 for both horizontal and vertical offsets, and the origin of low-level pattern is the same as the current composite pattern origin.

The **pitch** keyword followed by {*x_pitch y_pitch*} specifies the horizontal and vertical pitches when applying and repeating the low-level pattern *child_pattern_name* in the current composite pattern. The pitch is the distance between two adjacent low-level *child_pattern_name* patterns. By default, the pitch is 0 for both horizontal and vertical pitches and the low-level pattern is not repeated.

The **repeat** keyword followed by {*x_num y_num*} specifies the horizontal and vertical repeating numbers when duplicating the low-level pattern *child_pattern_name* in the current composite pattern. By default, the low-level pattern *child_pattern_name* is repeated horizontally and vertically by the specified pitches until the pattern reaches the routing area boundary.

You can specify multiple pattern expressions and separate them by curly brace pairs. For example, one pattern expression per low-level pattern is within a pair of curly braces. The index of the *i_th* low-level pattern added into this composite pattern is *i*.

-via_rule *via_rule_spec*

Specifies the via rule *via_rule_spec* between wires in this composite pattern. The via rule defines the intersection locations and the via at the locations. The rule will only apply between layers of composite pattern, but won't apply between layer of composite pattern and layer of other pattern. Users must define using set_pg_strategy_via_rule if control is needed between layer of composite pattern and layer of other pattern.

There are several methods to define the intersection locations in the *via_rule_spec* specification, where the syntax is defined as follows:

{**intersection** : all} {*via_def*} |
{**intersection** : adjacent} {*via_def*} |
list_of_filter_rules

The first method is "{**intersection** : all}", where **intersection** is the keyword and **all** refers to all intersections between any two orthogonal wires on any different layer.

The second method is "{**intersection** : adjacent}", where **intersection** is the keyword and **adjacent** refers to all intersections between any two orthogonal wires only between adjacent layers.

The third method is to define the intersections based on two sets of wire shapes using *list_of_filter_rules*. The rules contain a list of filtering rules. Each filtering rule is defined inside a curly brace pair. The syntax of each filtering rule is as follows:

{{*set_1*}{*filter_1*}} {{*set_2*} {*filter_2*}} {*via_def*} |
{**intersection** : undefined} {*via_def*}

where *set_1* refers to the first set of wire shapes and *filter_1* refers to the filtering operations for this set. *set_2* and *filter_2* refer to the second set of wire shapes and the filtering operations for the second set, respectively. *set_1* or *set_2* is defined as below:

{**pattern** : *pattern_name*}

where **pattern** is the keyword and *pattern_name* refers to one of the low-level patterns added to the current composite pattern by the **-add_patterns** option.

You can perform filtering operations on the set of wire shapes to obtained a subset of shapes. The filters are *filter_1* and *filter_2* in the previous example. The filtering operation is defined as follows:

{**net_id** : *n_id*} |
{**nets** : *net_list*} {**layers** : *layer_list*}
{**width** : {*lower_w upper_w*}

where **net_id** is the keyword and *n_id* refers to the ID of the specified low-level pattern. The index of the *i_th* net in *sub_net_list* is *i* when specifying the low-level pattern. The subset of wire shapes contains the wires with the specified *n_id*. Net ID is useful when there are multiple nets with the same names within the list of nets in the low-level pattern. Alternatively, **nets**, **layers**, **width** keywords can be used to specify the wire net names by *net_list*, wire layer names by *layer_list*, or a width range by {*lower_w upper_w*}. Only the wire shapes in the lower-level pattern which also satisfy the filtering categories form a subset of wire shapes. Here, *net_list* contains a list of net names, *layer_list* contains a list of layer names, and {*lower_w upper_w*} defines the lower and upper range of the wire width. By specifying two sets of wires by pattern names or pattern IDs and certain filtering operations by net IDs, net names, layer names or wire width ranges, the via locations are the intersections between these two sets of wire shapes. For those intersections which do not belong to any filtering rule, the via definition can be specified by

{**intersection** : undefined}{*via_def*}

where the **intersection** keyword followed by **undefined** refers to the remaining intersections which do not belong to any filtering rules. By default, the tool does not create vias at the intersections which do not belong to any filtering rules, unless specified. Multiple via filtering rules, such as intersection location and via definition, can be defined and separated by curly braces where each filter rule is in one curly brace pair.

With the intersection definition, the via specification {*via_def*} is in the following format:

{**via_master** : *via_master_list*}

where the **via_master** keyword followed by *via_master_list* specifies the list of via masters. Via masters include the via contact code defined in the technology file, or user-defined via cells. The **via_master** keyword followed by *via_rule_list* specifies a list of via master rules. Via master rules are defined by **set_pg_via_master_rule** command. **NIL** can be used as a keyword indicate that no via should be created. **default** can be used to describe the default vias in the specified location. For each specified intersection, vias are created based on the specified via masters or via master rules. If **NIL** is not specified in the list, the default contact code is used to complete a stacked via or to replace a via with DRCs when applicable.

By default, the default via defined in the technology file is created at each intersection of orthogonal wires in different layers.

DESCRIPTION

Creates a power ground composite pattern. This composite pattern can be defined hierarchically based on other low-level patterns such as the basic wire patterns using **create_pg_wire_pattern**, or other composite patterns using this command.

The via rules can also be defined between any wire shapes in this composite pattern. Each via rule contains the intersection location and the via definition by via master or via cell.

The composite pattern can be instantiated in a design by the **set_pg_strategy** command to create a power ground mesh.

EXAMPLES

The following example creates a composite pattern, checker_board, with one horizontal wire pattern hor_wire, one vertical wire pattern ver_wire, and the checker board via rule between two wire patterns. Note that net_id is used to specify the checker board via rule.

```
prompt> create_pg_composite_pattern checker_board -nets {VDD} \
  -parameters {ver_layer hor_layer ver_width hor_width ver_pitch hor_pitch} \
  -add_patterns { \
    {{name:ver_wire}{nets: {VDD VDD}} \
      {parameters: {@ver_layer @ver_width}}\
      {pitch: {0 @ver_pitch}} } \
    {{name:hor_wire}{nets: {VDD VDD}} \
      {parameters: {@hor_layer @hor_width}}\
      {pitch: {@hor_pitch 0}} } \
  } \
  -via_rule { \
    {{{pattern: ver_wire}{net_id: 1}} \
     {{pattern: hor_wire}{net_id: 1}}\
       {via_master: default}} \
    {{{pattern: ver_wire}{net_id: 2}}
     {{pattern: hor_wire}{net_id: 2}} \
       {via_master: default}} \
    {{{pattern: ver_wire}{net_id: 1}} \
     {{pattern: hor_wire}{net_id: 2}} \
       {via_master: NIL}} \
    {{{pattern: ver_wire}{net_id: 2}} \
     {{pattern: hor_wire}{net_id: 1}} \
       {via_master: NIL}} \
  }
```

The following example creates a composite pattern which contains wire pattern from layer M3 to M8, strap3, strap4, strap5, strap6 strap7 and strap8. The nets of the composite pattern is pwr and gnd, which are symbolic. The nets for each low-level wire pattern are in different order. The via rule is specified as adjacent, that is, the default via is created at each intersection between any orthogonal wires in adjacent layers.

```
prompt> create_pg_composite_pattern m3_to_m8 -nets {pwr gnd} \
  -add_patterns { \
    {{name:strap3}{nets:{gnd gnd pwr pwr}}{pitch:6.6} \
      {offset:{1.02 0}}}\
```

```
    {{name:strap4}{nets:{pwr gnd}}{pitch:{0 7.8}} \
      {offset:{0 5.9}}} \
    {{name:strap5}{nets:{gnd pwr}}{pitch:{6.6 0}} \
      {offset:{1.8 0}}} \
    {{name:strap6}{nets:{pwr gnd}}{pitch:{0 7.8}} \
      {offset:{0 5.9}}} \
    {{name:strap7}{nets:{gnd pwr}}{pitch:{14 0}}  \
      {offset:{1.8 0}}} \
    {{name:strap8}{nets:{gnd pwr}}{pitch:{0 9.6}} \
      {offset:{0 6.6}}} \
  } \
  -via_rule {{intersection: adjacent}{via_master: default}}
```

The following example creates a composite pattern composed of two wire patterns on layers M3 and M5. Vias with the via master Via_via35 are added between parallel wires M3 and M5.

```
set_pg_via_master_rule Via_via35 -contact_code {via3_CUST via4_CUST} \
  -via_array_dimension {2 1} \
  -allow_multiple {2.508 0.48} -cut_spacing {0.054 0.090}
create_pg_wire_pattern pg_wire -layer @l -direction @d -width @w \
  -pitch @p -spacing @s -parameters {l d w p s}
create_pg_composite_pattern pg_pat -nets VSS \
  -add_patterns { \
    {{pattern: pg_wire} {nets: VSS}
      {parameters: {M3 vertical {0.19 } 2.508 interleaving}}
      {offset: 02.508}} \
    {{pattern: pg_wire} {nets: VSS} \
      {parameters: {M5 vertical {0.23 } 2.508 interleaving}} \
      {offset: 02.508}} \
  } \
  -via_rule { \
    {{pattern_id: 1} {pattern_id: 2} \
      {via_master: Via_via35} {between_parallel: true}} \
      {{intersection: undefined} {via_master: NIL}} \
  }
```

In addition to the above examples, see the Examples page in the PG Planning section of the Task Assistant for more information. To view the Examples page, start the GUI and invoke the following command: **gui_show_task_assistant -task "Design Planning:PG Planning->Examples->Overview"**.

SEE ALSO

compile_pg(2)
create_pg_macro_conn_pattern(2)
create_pg_ring_pattern(2)
create_pg_std_cell_conn_pattern(2)
create_pg_wire_pattern(2)
remove_pg_patterns(2)
report_pg_patterns(2)
report_pg_strategies(2)
set_pg_strategy(2)
set_pg_via_master_rule(2)

create_pg_macro_conn_pattern

Creates a macro power ground connection pattern. Use the **set_pg_strategy** command to instantiate the pattern and connect macros to the power ground network.

SYNTAX

status **create_pg_macro_conn_pattern**
pattern_name
-pin_conn_type long_pin | ring_pin | scattered_pin
[-nets *net_list*]
[-parameters *var_list*]
[-direction horizontal | vertical | @*var*]
[-width *width* | {*hor_width ver_width*} | @*var*]
-layers *layer_name* | {*hor_layer ver_layer*} | @*var*
[-spacing minimum | interleaving | *spacing* |
{*hor_space ver_space*} | @*var*]
[-pitch *pitch* | {*hor_pitch ver_pitch*} | @*var*]
[-number *num* | {*hor_num ver_num*} | @*nvar*]
[-excluded_pins *pin_filter_spec*]
[-via_rule *via_rule_spec*]
[-pin_layers *layer_list* | @*var*]

Data Types

pattern_name string
net_list list
var_list list
var string
width float
hor_width float
ver_width float
layer_name string
hor_layer string
ver_layer string
spacing float
hor_space float
ver_space float
pitch float
hor_pitch float
ver_pitch float
num integer
hor_num integer
ver_num integer
nvar string
pin_filter_spec specification
via_rule_spec specification
layer_list list of string

ARGUMENTS

pattern_name

Specifies the name of the macro pattern.

-pin_conn_type long_pin | ring_pin | scattered_pin

Specifies the macro pin and connection types. This is a required option and you must specify the **long_pin**, **ring_pin** or **scattered_pin** keyword. The long_pin keyword refers to long pins in the macro; orthogonal straps will be created for connection. The ring_pin keyword refers to internal ring pins inside the macro; finger connections will be created to connect the ring pin to a strap or external macro ring in the external power network. The scattered_pin keyword refers to the irregular pin shapes scattered inside the macro; extension straps will be created to connect scattered pins to the external power network.

-nets *net_list*

Specifies power and ground net names used in this macro connection pattern. The net names are symbolic names and will be mapped from net names specified by using real net names in the **set_pg_strategy** command.

The net mapping is by position, that is, the *i_th* net name in **set_pg_strategy** command maps to the net name in the same position in this command. This option is not required and is needed when pin filtering based on net name is required. For example, use this option if certain nets can be skipped or pin connections in different nets have different via rules.

-parameters *var_list*

Specifies the list of parameters. Each parameter can be evaluated and used by other options in this command. This allows the pattern to be programmable and reused by passing different parameters in different situations. Each parameter *var* in the list *var_list* is a string and the at character (@) is used as a prefix to evaluate the parameter, such as *@var*. See the example section for more details.

-direction horizontal | vertical | *@var*

Specifies the direction of macro long pin connection straps. This option is required when you specify **-pin_conn_type long_pin**, otherwise, this option is ignored. This option specifies the direction for connection straps, not macro pins. The direction could be "horizontal", or "vertical", or specified by evaluating the variable *var* in the **-parameters** option. For parameters, the at character (@) is used as a prefix to evaluate the variable.

-width *width* | {*hor_width ver_width*} | *@var*

Specifies the width of the macro pin connection straps. If you specify **-pin_conn_type long_pin**, only one width value is required. Otherwise, the horizontal width and vertical width must be specified with *hor_width* and *ver_width*, respectively. The unit is defined in the technology file.

The width can also be specified by evaluating the variable *var* in **-parameters** option. The at character (@) is used as a prefix to evaluate the variable. By default, the minimum width defined in the technology file is used if this option is not specified.

-layers *layer_name* | {*hor_layer ver_layer*} | *@var*

Specifies the layer of the macro pin connection straps. This is a required option. If you specify **-pin_conn_type long_pin**, only one layer *layer_name* is required. Otherwise, the horizontal and vertical layers must be specified with *hor_layer* and *ver_layer*, respectively.

The layer names can be also specified by evaluating the variable *var* in **-parameters** option. The at character (@) is used as a prefix to evaluate the variable.

-spacing minimum | interleaving | *value* | {*hor_space ver_space*} | *@var*

Specifies the spacing between macro connection straps in the pattern. This option only applies if you specify **-pin_conn_type long_pin** or **-pin_conn_type ring_pin**. If you specify **-pin_conn_type scattered_pin**, this option is ignored.

minimum keyword refers to the minimum spacing between connection straps defined in the technology file, based on the wire width. **interleaving** keyword means that the distance between adjacent center lines of connection straps is equal.

Specific values can also be used for spacing. If you specify **-pin_conn_type long_pin**, only one *value* is required. If you specify **-pin_conn_type ring_pin**, the horizontal and vertical spacing values must be specified with *hor_space* and *ver_space*, respectively.

The spacing can also be specified by evaluating the variable *var* in **-parameters** option. The at character (@) is used as a prefix to evaluate the variable. By default, the minimum spacing is used if this option is not specified.

-pitch *pitch* | {*hor_pitch ver_pitch*} | @*var*

Specifies the pitch of macro connection straps in the pattern. The pitch only applies if you specify **-pin_conn_type long_pin** or **-pin_conn_type ring_pin**. If the connection type is **scattered_pin**, the pitch is ignored.

If the connection type is **long_pin** or **ring_pin**, at least one of the **-pitch** or **-number** options must be specified. If both options are specified, both options are honored. Otherwise, one value is derived from another by evenly distributing the connection straps in the macro.

If the pin connection type is **long_pin**, one *value* must be specified for the pitch value. If the type is **ring_pin**, horizontal and vertical pitch values must be specified by *hor_pitch* and *ver_pitch*, respectively. The unit is defined in the technology file. The pitch values can also be specified by evaluating the variable *var* in the **-parameters** option. The at character (@) is used as a prefix to evaluate the variable.

-number *num* | {*hor_num ver_num*} | @*var*

Specifies the repeating number of macro connection straps in the pattern. It only applies if you specified **-pin_conn_type long_pin** or **-pin_conn_type ring_pin**. If the connection type is **scattered_pin**, the number is ignored.

If the connection type is **long_pin** or **ring_pin**, at least one of **-pitch** or **-number** option must be specified. If both are specified, both are honored. Otherwise, one value is derived from another by evenly distributing the connection straps in the macro.

If the pin connection type is **long_pin**, one *num* must be specified for the repeating pattern. If the type is **ring_pin**, horizontal and vertical numbers must be specified by *hor_num* and *ver_num*, respectively. The repeating number can also be specified by evaluating the variable *var* in **-parameters** option. The at character (@) is used as a prefix to evaluate the variable.

-excluded_pins *pin_filter_spec*

This option is not required. This option only applies when you specify **-pin_conn_type scattered_pin**.

For **scattered_pin** connection type, certain pin shapes that satisfy the *pin_filter_spec* specification can be excluded for connection. The *pin_filter_spec* specification uses the following format:

{**layers** : *layer_list*}

where **layers** is the keyword for pin layers. *layer_list* is a list of strings which contains a list of layer names. The pin shapes which satisfy the above criteria are excluded from connection. By default, all pin shapes are connected if this option is not specified.

-via_rule *via_rule_spec*

Specifies the via rule *via_rule_spec* between connection straps and macro pins in this macro connection pattern. The via rule defines the intersection locations and the via at the locations.

There are several ways to define the intersection locations in the specification *via_rule_spec*, where the syntax is defined as follows:

{**intersection** : all} {*via_def*} |
{**intersection** : adjacent} {*via_def*} |

list_of_filter_rules |

The first method is "{**intersection** : all}", where **intersection** is the keyword and **all** refers to all intersections between any two connection strap and macro pin across any layers.

The second method is "{**intersection** : adjacent}", where **intersection** is the keyword and **adjacent** refers to all intersections between any two connection strap and macro pin only in adjacent layers.

With the intersection definition, the via specification {*via_def*} uses the following format:

{**via_master** : *via_master_list*}

where **via_master** is the keyword for via masters including via contact code defined in the technology file or user-define via cells, or the PG via rules defined by **set_pg_via_master_rule** command. The *via_master_list* specifies the list of via masters. Via masters are created at the intersection center without replication. If offset or replication is needed, via rules specified by **set_pg_via_master_rule** command can be used. The *via_master_list* can also contain a list of PG via rules. **NIL** can be used as a keyword to indicate that no via is created. **default** can be used to describe the default vias in the specified location. If **NIL** is not specified in the list, the default contact code is used to complete a stacked via or to replace a via with DRCs when applicable.

By default, the default via defined in the technology file is created at each intersection of orthogonal connection straps and macro pins in different layers.

-pin_layers ***layer_list*** | ***@var***

Specifies a list of layers for allowed pin connections. Only pins on the specified layers can be used for a PG connection. If not specified, pins on all layers will be considered for a PG connection.

DESCRIPTION

Creates a macro power ground connection pattern. This pattern can be instantiated in a design with the **set_pg_strategy** command to connect macros to the power ground network. There are three types of macro connections: **long_pin**, **ring_pin** and **scattered_pin**. This pattern can be also applied to power pad connection.

EXAMPLES

The following example creates a macro connection pattern with long pin type. The pattern name is long_pin_pattern, the connection strap direction is horizontal, the layer is M5, symbolic net names are pwr and gnd respectively and width, spacing, pitch are specified through parameters w, s and p respectively. Default vias are created at all intersections between connection straps and macro pins.

```
prompt> create_pg_macro_conn_pattern long_pin_pattern \
  -pin_conn_type long_pin -nets {pwr gnd} \
  -direction horizontal -width @w \
  -layers M5 -spacing @s -pitch @p \
  -parameters {w s p}
```

The following example creates a macro connection pattern with ring pin type. The pattern name is ring_pin_pattern. The horizontal and vertical layers are M5 and M6 respectively. Width, spacing, pitch are specified by parameters.

```
prompt> create_pg_macro_conn_pattern ring_pin_pattern \
  -pin_conn_type ring_pin \
  -width {@hor_w @ver_w} -layers {M5 M6} \
```

```
    -spacing {@hor_s @ver_s} -pitch {@hor_p @ver_p} \
    -parameters {hor_w ver_w hor_s ver_s hor_p ver_p}
```

The following example creates a macro connection pattern with scattered pin connection type. The pattern name is scatter_pin_pattern. The connection strap width is specified with parameters. Horizontal and vertical layers are M5 and M6 respectively. Default vias are created between macro pins and connection straps.

```
prompt> create_pg_macro_conn_pattern scatter_pin_pattern \
  -pin_conn_type scattered_pin -nets {pwr gnd} \
  -width {@hor_w @ver_w} -layers {M5 M6} \
  -parameters {hor_w ver_w}
```

MORE EXAMPLES

For more example, please start GUI and invoke the following command. **gui_show_task_assistant -task "Design Planning:PG Planning->Examples->Overview"**.

SEE ALSO

compile_pg(2)
create_pg_composite_pattern(2)
create_pg_ring_pattern(2)
create_pg_std_cell_conn_pattern(2)
create_pg_wire_pattern(2)
remove_pg_patterns(2)
report_pg_patterns(2)
report_pg_strategies(2)
set_pg_strategy(2)
set_pg_via_master_rule(2)

create_pg_mesh_pattern

Creates a power ground mesh pattern. The mesh pattern can be instantiated in a design by **set_pg_strategy** command to create power ground mesh.

SYNTAX

status **create_pg_mesh_pattern**
pattern_name
-layers *layer_expr*
[-parameters *var_list*]
[-via_rule *via_rule_spec*]

Data Types

pattern_name string
layer_expr specification
var_list list
via_rule_spec specification

ARGUMENTS

pattern_name

Specifies the name of the mesh pattern.

-parameters *var_list*

Specifies the list of parameters. Each parameter can be evaluated and used by other options in this command. This allows the pattern to be programmable and reused by passing different parameters in different situations. Each parameter *var* in the list *var_list* is a string and the at character (@) is used as a prefix to evaluate the parameter, such as *@var*. See the example section for more details.

-layers *layer_expr*

Specifies the layer configuration for this mesh pattern. Each layer is specified by a *layer_expr* expression in the following syntax:

{horizontal_layer | vertical_layer: *layer_name* | *@var*}
{width: minimum | *value_list* | *@var*}
{spacing: interleaving | minimum | *value_list* | *@var*}
{offset: *offset_value* | *@var*}
{pitch: *pitch_value* | *@var*}
{trim : true | false | *@var*}
{track_alignment : track | half_track | auto | *@var*}
{mask : mask_one | mask_two | mask_three | mask_one_mask_two_alternate |
mask_two_mask_one_alternate | *@var*}

{mask_constraint : *constraint_name*}

where the specification contains several keywords, **horizontal_layer** or **vertical_layer**, **width**, **offset**, **pitch**, **spacing**, **trim**, **mask** and **mask_constraint**.

The **horizontal_layer** or **vertical_layer** keyword specifies the layer name for this layer. Only one of **horizontal_layer** and **vertical_layer** can be specified within one layer. The layer name is specified by *layer_name* or by evaluating the variable *var* in **-parameters** option. The at character (@) is used as a prefix to evaluate the variable. The layer name is a required option in the layer spec.

The **width** keyword specifies the width of this layer. "minimum" refers to the minimum wire width defined in the technology file. *value_list* contains a list of width values, where the width value is a float-type value with units defined in the technology file. If the number of width values is smaller than the number of nets from the **set_pg_strategy** command, the last width value is assumed for the extra nets. The width can also be specified by evaluating the *var* variable in **-parameters** option. The at character (@) is used as a prefix to evaluate the variable. By default, the minimum width is used if this option is not specified.

The **offset** keyword specifies the offset value. *offset_value* is the offset applied to the center line of the first wire to the boundary of the routing region or a specified absolute coordinate in **set_pg_strategy** command. By default, the offset is 0. The offset can also be specified by evaluating the *var* variable in **-parameters** option.

The **pitch** keyword specifies the pitch value. *pitch_value* specifies the pitch of the wires in this layer. The pitch can also be specified by evaluating the *var* variable in **-parameters** option. This pitch is a required option in the layer spec.

The **spacing** keyword specifies the spacing between wires in this layer. **minimum** refers to the minimum spacing between wires defined in the technology file based on the wire width. *value_list* contains a list of spacing values, where the spacing value is a float-type value with unit defined in the technology file. Each spacing value specifies the spacing between the adjacent wires in the pattern. For example, the first spacing value specifies the spacing between the first and the second wires, and so on. The number of wires in this pattern is specified by the **set_pg_strategy** command. If the number of spacing values is smaller than the number of nets minus one, the last spacing value is assumed for the extra nets. **interleaving** keyword means that distance between adjacent center lines of wires is equal. The spacing can also be specified by evaluating the *var* variable in **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable. By default, the minimum spacing is used if this option is not specified.

The **trim** keyword specifies the trim option for wires in this layer. Valid values are true, false or a value specified by evaluating the *var* variable in **-parameters** option. By default, the trim option is true.

The **track_alignment** keyword specifies the track alignment option for wires in this layer. The option can be track, half_track, or auto. When auto is specified, tool choose the alignment method with smaller number of used tracks. It can also be specified by evaluating the *var* variable in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable. By default, the wires are created at the specified location and no alignment is performed.

The **mask** keyword specifies the mask option for wires in this layer. The mask can be specified by one of the keywords, mask_one, mask_two, mask_three, mask_one_mask_two_alternate or mask_two_mask_one_alternate. It can also be specified by evaluating the *var* variable in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable. By default, the wires are not colored with mask if not specified.

The **mask_constraint** keyword specifies the mask constraint option for wires in this layer. The mask constraint is defined by the **set_pg_mask_constraint** command. It can also be specified by evaluating the *var* variable in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable. By default, the wires are not checked against mask constraint.

Multiple layer expressions can be specified and separated by brace pairs, that is, one layer expression per layer is within a pair of braces.

-via_rule ***via_rule_spec***

Specifies the *via_rule_spec* via rule between wires in this mesh pattern. The via rule defines the intersection locations and the via at the locations. The rule will only apply between layers of mesh pattern, but won't apply between layer of mesh pattern and layer of other pattern. You must define via rules with the **set_pg_strategy_via_rule** command if control is needed between layer of mesh pattern and layer of other pattern.

There are several ways to define the intersection locations in the *via_rule_spec* specification, where the syntax is defined as follows:

{**intersection** : all} {*via_def*} |
{**intersection** : adjacent} {*via_def*} |
list_of_filter_rules |

The first method is "{**intersection** : all}", where **intersection** is the keyword and **all** refers to all intersections between any two orthogonal wires in any different layers.

The second method is "{**intersection** : adjacent}", where **intersection** is the keyword, and **adjacent** refers to all intersections between any two orthogonal wires only in adjacent layers.

The third method is to define the intersections based on two sets of wire shapes using *list_of_filter_rules*, which contains a list of filtering rule. Each filtering rule is inside a brace pair. The syntax of each filtering rule is as follows:

{*filter_1*}{*filter_2*}{*via_def*} |
{**intersection** : undefined}{*via_def*}

where *filter_1* refers to the first filtering operation and *filter_2* refer to the second filtering operation. The intersection is defined between the first set and second set of wires based on the filtering rules. The second filtering rule is optional, that is, the intersection is defined based on the first set of wires to all other wires in this mesh.

The filtering operation is defined as follows:

{**layers** : *layer_list*} {**width** : {*lower_w upper_w*}}

where the **layers** and **width** keywords can be used to specify the wire layer names by *layer_list* or width range by {*lower_w upper_w*}. Here, *layer_list* contains a list of layer names, and {*lower_w upper_w*} is the lower/upper range of wire width. For those intersections which do not belong to any filtering rule, the via definition can be specified as follows:

{**intersection** : undefined}{*via_def*}

where **intersection** is the keyword and **undefined** refers to the remaining intersections which do not belong to any filtering rules. By default, vias are only created at the intersections which belong to a filtering rule. Multiple via filtering rules for intersection location and via definition can be defined and separated by braces where each filter rule is in one brace pair.

With the intersection definition, the via specification {*via_def*} uses the following format:

{**via_master** : *via_master_list* | *via_rule_list*}

where **via_master** is the keyword for via masters including via contact code defined in the technology file or user-defined via cells, The PG via rules defined by **set_pg_via_master_rule** command can be also specified in the list. *via_master_list* specifies the list of via masters. Via masters are created at the intersection center without replication. If offset or specific array dimension is needed, via rules specified by **set_pg_via_master_rule** command can be used. *via_rule_list* is a list of PG via rules. **NIL** can be used as a keyword indicate that no via would be created. **default** can be used to describe the default vias in the specified location. Multiple via rules can be specified and separated by brace pairs. If **NIL** is not specified in the list, default contact code will be used to complete a stacked via or to replace a via with DRCs when applicable.

By default, the default via defined in the technology file will be created at each intersection of orthogonal wires in different layers.

DESCRIPTION

Creates a power ground mesh pattern. This mesh pattern can be used to create PG mesh with the **set_pg_strategy** command.

The via rules can also be defined based on intersection or filtering rules. Each via rule contains the intersection location and the via definition by contact code or via master rule.

EXAMPLES

The following example creates a mesh pattern with three layers. Parameters are used for width and offset. The vias between adjacent layers are specified in the via rule. **set_pg_strategy** and **compile_pg** are used to create the mesh.

```
prompt> create_pg_mesh_pattern mesh \
  -parameters {w1 w2 w3 f} \
  -layers { \
    {{vertical_layer: M8}{width: @w1} \
       {spacing: interleaving}{pitch: 20} {offset: @f}} \
    {{horizontal_layer: M7}{width: @w2} \
       {spacing: interleaving} {pitch: 20} {offset: @f}} \
    {{vertical_layer: M6}{width: @w3} \
       {spacing: interleaving}{pitch: 20} {offset: @f}} \
  } \
  -via_rule { \
    {{layers: M8}{layers: M7}{via_master: VIA78}} \
    {{layers: M7}{layers: M6}{via_master: VIA67}} \
  }

prompt> set_pg_strategy smesh \
  -pattern { \
    {pattern: mesh}{nets: VDD VSS} \
       {parameters: 5 3 1 3}} \
  -core

prompt> compile_pg
```

The following example creates a mesh pattern with layers with same direction. Parameters are used for width and offset. The vias between parallel layers M8 and M6 are specified in the via rule.

```
prompt> create_pg_mesh_pattern mesh \
  -parameters {w1 w3 f} \
  -layers { \
    {{vertical_layer: M8}{width: @w1} \
       {spacing: interleaving}{pitch: 20} {offset: @f}} \
    {{vertical_layer: M6}{width: @w3} \
       {spacing: interleaving}{pitch: 20} {offset: @f}} \
  } \
  -via_rule { \
    {{layers: M8}{layers: M6} \
      {via_master: VIA78 VIA67}{between_parallel: true}} \
  }
```

In addition to the above examples, see the Examples page in the PG Planning section of the Task Assistant for more information. To view the Examples page, start the GUI and invoke the following command: **gui_show_task_assistant -task "Design Planning:PG Planning->Examples->Overview"**.

SEE ALSO

compile_pg(2)
create_pg_macro_conn_pattern(2)
create_pg_ring_pattern(2)
create_pg_std_cell_conn_pattern(2)
create_pg_wire_pattern(2)
remove_pg_patterns(2)
report_pg_patterns(2)
report_pg_strategies(2)
set_pg_strategy(2)
set_pg_via_master_rule(2)
set_pg_mask_constraint(2)

create_pg_ml_data

Create ML training files based on created ML training data in memory.

SYNTAX

status **create_pg_ml_data**
[-output_directory *directory_name*]

Data Types

directory_name string

ARGUMENTS

-output_directory *directory_name*

Specify directory name for created ML training file. The option is required.

DESCRIPTION

This command creates ML training files based on created ML training data in memory.

EXAMPLES

The following example creates ML training data in memory, output training data to files in directory pg_data_1, and train ML model based on the training data. The trained ML model is stored in the directory ./pg_model.

prompt> **compile_pg -create_ml_data**
prompt> **create_pg_ml_data -output_directory pg_data_1**
prompt> **train_pg_ml_model -input_directory {pg_data_1}**

SEE ALSO

compile_pg(2)
create_pg_vias(2)

create_pg_pattern_shapes

Creates power ground wire shapes in pattern-based. Vias will not be created in the command.

SYNTAX

status **create_pg_pattern_shapes**
-layer *layer_name*
-net *net_name*
-width *value*
-low_end *value*
-high_end *value*
-start *value*
-direction horizontal | vertical
[-max_array_size *number*]
[-within_bbox { {*llx lly*} {*urx ury*} }]
[-xPitch *value*]
[-yPitch *value*]
[-ignore_drc]
[-mark_as *pg_type*]
[-tag *tag_name*]
[-mask mask_one | mask_two | mask_three]
[-mask_pattern uniform | alternate_row | alternate_column]
[-blockage {*blockage_spec*}]

Data Types

layer_name string
net_name string
value float
number integer
llx float
lly float
urx float
ury float
tag_name string
pg_type specification
blockage_spec specification

ARGUMENTS

-layer *layer_name*

Specifies the layer name on which to create the PG pattern shapes. This is a required option.

-net *net_name*

Specifies net name of the PG pattern shapes by the string *net_name*. This is a required option.

-width *value*

Specifies the width of repeating PG shapes in the PG patterns shapes. This is a required option. *value* is a float value with um as the unit.

-low_end *value*

Specifies the low end coordinate of lower-left PG shapes in the PG patterns shapes. It could be specified as a fixed *value* with um as the unit. This is a required option.

-high_end *value*

Specifies the high end coordinate of lower-left PG shapes in the PG patterns. It could be specified as a fixed *value* with um as the unit. This is a required option.

-start *value*

Specifies the starting position of center-line of repeating PG shapes in the PG patterns shapes. This is a required option. *value* is a float value with um as the unit.

-direction horizontal | vertical

Specifies the direction of repeating PG shapes in the PG patterns shapes. This is a required option. Legal directions are "horizontal" and "vertical".

-max_array_size *number*

Specifies the max array size of of one PG pattern shapes. The number limit max repeated size in both two dimension. This option is not required. If not specified, the default value is 1000.

-within_bbox { {*llx lly*} {*urx ury*} }

Specifies the region that pattern shapes should be created. This option is not required. If not specified, pattern shapes will be created in corea area.

-xPitch *value*

Specifies the pitch in X axis. It will be x pitch in one pattern shapes. It could be specified as a fixed *value* with um as the unit. This option is not required. If the option is not specified, shape will not be repeated horizontally.

-yPitch *value*

Specifies the pitch in Y axis. It will be y pitch in one pattern shapes. It could be specified as a fixed *value* with um as the unit. This option is not required. If the option is not specified, shape will not be repeated vertically.

-ignore_drc

Specifies this option for PG pattern shapes creation. If speficied, DRC will be ignored. The PG pattern shapes will be created, based on the specification when this option is used. By default, DRC is activated and fixed for the PG pattern shapes.

-mark_as *pg_type*

Specifies the type of the PG pattern shapes to be created. The argument must by on of these keywords: stripe, ring, lib_cell_pin_connect, macro_pin_connect, user_route and follow_pin. These types refer to stripe, ring, standard cell connection, macro connection, user_route and follow_pin respectively. By default, the stripe type is used.

-tag *tag_name*

Specifies a tag name to assign to all new shapes created by this command. This tag name is used as contents of an user attribute tag for filtering ollections at later stages (see the example in the EXAMPLES section). By default, no tags are assigned.

-mask mask_one | mask_two | mask_three

Specifies the mask option for the PG pattern shapes. By default, the PG pattern shapes is not colored with a mask identifier. The type of mask can be specified by one of the keywords, mask_one, mask_two, mask_three.

-mask_pattern uniform | alternate_row | alternate_column

Specifies the mask option for the PG pattern shapes. By default, the PG pattern shapes mask pattern is uniform. The type of mask can be specified by one of the keywords, uniform, alternate_row, alternate_column.

-blockage {*blockage_spec*}

Specifies the routing blockage in the power ground network. You can specify multiple blockages within the braces, one blockage per group. Blockages are separated by braces and contain the keywords **nets:**, **layers:**, and a target specification as follows:

{{nets: *nets*} {layers: *layers*} {*target*}

where *target* is a voltage area, block, macro, macro with hard keep-out-margin (KOM), PG region, hard placement blockage, or polygon that is specified as follows:

{voltage_areas : *voltage_areas* | macros : *macro_names* |
macros_with_keepout : *macro_names* | placement_blockages : all |
polygon : {*polygon_area*} | blocks : *blocks* |
pg_regions : *region_list* }

The net names following the *nets* keyword are a subset of the nets specified by **-nets** in *pattern_expr* from **-pattern** option. If you do not specify the **nets** keyword, the blockage applies to all the nets in the strategy. The layer names following the **layers** keyword are the routing layers on which power or ground straps are blocked. If you do not specify the **layers** keyword, the blockage applies to all routing layers. Use one of the **voltage_areas**, **macros**, **polygon**, **blocks** or **pg_regions** keywords to represent the blockage area. *polygon_area* is specified as a list of coordinate points.

DESCRIPTION

Creates power ground wire shapes in pattern-based. Vias will not be created in the command. The command will create one shape pattern object that can represent repeating PG shapes based on user specified spec. DRC engine will check each element in pattern shapes. Fixing engine will modify the shape by cutting or remove single wire shape. For the fixed wire shapes, the geometry will not be the same as input. It will generate non-repeating PG shapes. Non-repeating PG shapes will be created as normal wire shapes.

EXAMPLES

The following example creates a VDD PG pattern shapes on layer M3, center at 0.993(X-axis), width as 0.037um, low end at 1(Y-axis), high end at 1.3(Y-axis) and vertical direction. Each wires shapes repeated every 4.959 um in x direction. -yPitch is not specified, so will created one row of vertical wire shapes. The shapes in the pattern will mask lower-left shape as mask one. The color will alternate every other column.

```
prompt> create_pg_pattern_shapes \
  -width 0.037 \
  -net VDD \
```

```
-low_end 1 -high_end 1.3 \
-start 0.993 -xPitch 4.959 \
-direction vertical \
-layer M3 \
-mark_as stripe \
-mask_pattern alternate_column -mask {mask_one}
```

SEE ALSO

compile_pg(2)
create_pg_vias(2)
create_pg_strap(2)

create_pg_region

Defines a PG region used to create a power ground network. The region can be used as a routing region or a blockage constraint for PG creation.

SYNTAX

status **create_pg_region**
region_name
[-core | -design_boundary |
-group_of_macros *macro_cells* |
-group_of_ios *io_cells* |
-block *block* |
-polygon {{ {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } |
geometric_objects} |
-voltage_area *voltage_area* |
-join_regions *region_list* |
-update *old_name*]
[-expand {*h_offset v_offset*}]
[-expand_by_edge {*edge_offset_spec*}]
[-exclude_regions *region_list*]
[-exclude_macros *macro_cells*]
[-macro_offset {*x_offset y_offset*}]
[-io_offset {*x_offset y_offset*}]
[-remove_jog {*jog_spec*}]
[-remove_notch {*notch_spec*}]

Data Types

region_name string
macro_cells collection or list
io_cells collection or list
block collection or list of one object
llx float
lly float
urx float
ury float
x float
y float
geometric_objects collection
voltage_area collection or list of one object
region_list list
old_name string
x_offset float
y_offset float
h_offset float
v_offset float

edge_offset_spec specification
jog_spec specification
notch_spec specification

ARGUMENTS

region_name

Specifies the name of the power ground region. This option is required.

-core

Defines the power ground region as the core area boundary. The command saves the region with the name *region_name*.

-design_boundary

Defines the power ground region as the design boundary. The command saves the region with the name *region_name*.

-group_of_macros {*macro_cells*}

Defines the power ground region as the contour around the specified macros. The command saves the region with the name *region_name*.

-group_of_ios {*io_cells*}

Defines the power ground region as the contour around the specified ios. The command saves the region with the name *region_name*.

-block *block*

Defines the power ground region as the contour around the specified soft macro block. You can specify a single block only. The command saves the region with the name *region_name*.

-polygon { {*llx lly*} {*urx ury*} } | {{*x y*} {*x y*} {*x y*} {*x y*}...} | *geometric_objects*

Defines the power ground region as the specified polygon area. The command saves the region with the name *region_name*.

The polygon area may be specified as a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates (i.e., {*llx lly*} {*urx ury*}). A polygon is specified by its points (i.e., {*x y*} {*x y*} {*x y*} {*x y*}...).

A polygon may also be specified as the combined area of a heterogenous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

When specifying the polygon area as a collection of physical objects, the resulting area must resolve to a single, connected polygon. It is an error to specify a collection of objects with a combined area that resolves to multiple polygons.

-voltage_area *voltage_area*

Defines the power ground region as the specified voltage area. You can specify a single voltage area only. The command saves the region with the name *region_name*.

-join_regions *region_list*

Defines the power ground region as the union of the contours of the specified regions. If the regions do not touch or overlap, the command expands the new region to form a contiguous boundary. The command saves the region with the name *region_name*.

-update *old_name*

Copies the power plan region *old_name* and modifies it by applying operations specified by other options. The command saves the new region with the name *region_name*.

The options **-core**, **-design_boundary**, **-voltage_area**, **-polygon**, **-block**, **-group_of_macros**, **-join_regions**, and **-update** are mutually exclusive. You can specify only one of these options.

You can perform operations on the defined region, including expanding or shrinking by horizontal and vertical offsets, excluding certain macros with offset, excluding a list of previously defined power ground regions, smoothing out narrow channels between macros, and so on. Not all operations are supported for all regions.

If you specify more than one modification operation, the tool applies the operations in the following sequence: expand the region, exclude regions, exclude macros, remove notches and jogs. You can change this order by running multiple **create_pg_region -update** commands and specifying one modification at a time.

-expand {*h_offset v_offset*}

Specifies the horizontal and vertical expansion offsets for the region. The offset unit is um. *h_offset* is applied to horizontal edges and *v_offset* is applied to vertical edges. You can specify negative offsets to shrink the region. By default, the offset is 0 and no region adjustment is performed.

-expand_by_edge *edge_offset_spec*

Specifies the expansion option for edges. Edge could be specified by either direction or an index. The supported directions are left, right, top, bottom, leftmost, rightmost, topmost, and bottommost. When edges are specified by index, each edge is indexed by a unique number. Edge numbering starts from 1 at the leftmost vertical edge. If there are more than one leftmost edges, the bottommost edge is the starting edge. The edge number increases by 1 as you proceed around the shape in the clockwise direction. *edge_offset_spec* specifies the offset for each edge as a pair of **edge_id** and **offset** keywords. An example is as follows:

{{side: *edge_id*}{offset: *edge_offset*}}

where *edge_id* and *edge_offset* specify the edge id and edge offset respectively. The offset unit is um. You can specify negative offsets to shrink the region. If you do not specify offsets for all edges, the offset for the remaining edges is either 0 or the uniform offset for all edges set by the **-expand** option. By default, the offset is 0 for all edges and no region adjustment is performed.

-exclude_regions *region_list*

Specifies a list of power ground region names to exclude from the current region. You can use the **-exclude_regions** option together with the **-core**, **-voltage_area**, **-polygon**, **-group_of_macros** and **-update** options. You cannot use **-exclude_regions** together with the **-join_regions** options. By default, no regions are excluded.

If you use the **-expand** or **-expand_by_edge** and **-exclude_regions** options together, the tool first expands the region, then excludes regions by default. To change the default execution order, use the **-update** option with one option at a time.

-exclude_macros *macro_cells*

Specifies a list or collection of hard macros to exclude from the power ground region. You can use the **-exclude_macros** option together with the **-core**, **-voltage_area**, **-polygon**, **-join_regions** and **-update** options. You cannot use **-exclude_macros** together with the **-group_of_macros** options. By default, no hard macros are excluded.

If you use the **-expand** or **-expand_by_edge** and **-exclude_macros** options together, the tool first expands the region, then excludes macros by default to honor the offset specified for the macros. To change the execution order of macro expansion and exclusion, use the **-update** option with one option at a time.

-macro_offset {*x_offset y_offset*}

Specifies the horizontal and vertical edge offsets for the list or collection of hard macros to be excluded from the region. This option only takes effect when used together with the **-exclude_macros** option or the **-group_of_macros** option. The offset is uniformly applied to all specified hard macros. The offset unit is um. By default, the offset is 0.

-io_offset {*x_offset y_offset*}

Specifies the horizontal and vertical edge offsets for the list or collection of ios to be excluded from the region. This option only takes effect when used together with **-group_of_ios** option. The offset is uniformly applied to all specified ios. The offset unit is um. By default, the offset is 0.

-remove_jog *jog_spec*

Specifies the method used to remove jogs and jog threshold. The **expand** keyword specifies that the region is expanded to remove jogs. The **shrink** keyword specifies that the region is shrunk to remove jogs. Any jog edge in the region shorter than the threshold specified by the *threshold* value is removed by expanding or shrinking the current region. The syntax of *jog_spec* is as follows:

{**expand** | **shrink** : *threshold*}

where only one of **expand** and **shrink** can be specified and *threshold* is a floating value with unit as um. If you specify the **-group_of_macros** or **-join_regions** option, the tool removes jogs by expanding the region automatically. If you specify the **-exclude_macros** or **-exclude_regions** option, the tool removes jogs by shrinking the region automatically. Otherwise, you must specify the removal method by using the one of **expand** and **shrink** keywords.

-remove_notch *notch_spec*

Specifies the notch type to be removed from the region and the notch threshold. The **convex** keyword specifies that the tool removes **convex** notches inside the region. The **concave** keyword specifies that the tool removes concave notches outside the region. The tool removes convex notches by shrinking the region, and removes concave notches by expanding the region. The tool only removes any convex or concave notches with a width smaller than the specified *threshold*. The syntax of *notch_spec* is as follows:

{**convex** | **concave** : *threshold*}

where only one of **convex** and **concave** can be specified and *threshold* is a floating value in um. If you also specify the **-group_of_macros** or **-join_regions** option, the tool removes concave notches by expanding the region automatically. If you specify the **-exclude_macros** or **-exclude_regions** option, the tool removes convex notches by shrinking the region automatically. Otherwise, you must specify the notch type by using one of **convex** and **concave** keywords.

DESCRIPTION

This command defines the regions used to create the power ground network. The region can be used as a routing region or a blockage for power mesh or ring creation.

You can specify the region by core area, by voltage area boundary, by polygon area, by contour of the macro groups, by contour from joining a list of previously defined power ground regions, or by incrementally modifying a previous defined region. You can perform operations on the defined region, including expanding or shrinking by horizontal and vertical offsets, excluding certain macros with offset, excluding a list of previously defined power ground regions, smoothing out narrow channels between macros, and so on. Not all operations are supported for all regions.

If you specify more than one operation, the tool runs the operations in a specific sequence. The tool begins by expanding the region, followed by excluding regions, excluding macros, and notch or jog removal is the last step. You can change the order by specifying the **-update** option with one operation at a time.

The power ground region only supports simple polygons and does not support donut-shape, self-cut or disjoint polygons.

EXAMPLES

The following example defines a power ground region with the name pp_region1. The region is specified with the coordinates {{1200 952} {1200 650} {1400 650} {1400 952}}.

```
prompt> create_pg_region pp_region1 \
  -polygon {{1200 952} {1200 650} {1400 650} {1400 952}}
```

The following example defines a region with the name *pp_region2* based on the core area boundary. The core area is first shrunk by 50 um. The $macro1 and $macro2 hard macros are excluded from the region. A horizontal edge offset of 20 um and vertical edge offset of 30 um are applied to both hard macros. Any notch areas inside the region with width smaller than 30 um are removed.

```
prompt> create_pg_region pp_region2 \
  -core -expand -50 \
  -exclude_macros "$macro1 $macro2" \
  -macro_offset {20 30} \
  -remove_notch 30
```

The following example defines a region named *new_region* that is copied from the previously defined region *old_region*. The *old_region* region is copied and expanded by 200 um to create the new region. The concave notches outside the region with width smaller than 10 um are removed by expanding the region. The jog edges shorter than 10 um are removed by expanding the region.

```
prompt> create_pg_region new_region \
  -update old_region -expand 200 \
  -remove_notch {concave: 10} \
  -remove_jog {expand: 10}
```

The following example defines a region named *pp_region3* that is based on the contour of the three macros $macro1, $macro2 and $macro3. The uniform horizontal and vertical offset applied to both hard macros is 20 um. The notches outside the region with width smaller than 30 um are removed by expanding the region. The jog edges shorter than 10 um are removed as well.

```
prompt> create_pg_region pp_region3 \
  -group_of_macros "$macro1 $macro2 $macro3" \
  -macro_offset {20 20} \
  -remove_notch 30 -remove_jog 10
```

The following example defines a region with the name *pp_region4* based on the voltage area boundary of va_top and expanded by 30 um horizontally and 20 um vertically.

```
prompt> create_pg_region pp_region4 \
  -voltage_area va_top -expand {30 20}
```

The following example defines a region with the name *pp_region5* based on core area and expanded by 10 um, 20 um, 30 um, and 40 um for the four edges assuming that the core is a rectangle.

```
prompt> create_pg_region pp_region5 \
  -core \
  -expand_by_edge { \
    {{side: 1}{offset: 10}} {{side: 2}{offset: 20}} \
    {{side: 3}{offset: 30}} {{side: 4}{offset: 40}} \
  }
```

The following example defines a region with the name *pp_region5* based on core area and expanded by 10 um, 20 um, 30 um, and 40 um for edges at left, top, right, and bottom.

```
prompt> create_pg_region pp_region6 \
```

```
-core \
-expand_by_edge { \
  {{side: left}{offset: 10}} {{side: top}{offset: 20}} \
  {{side: right}{offset: 30}} {{side: bottom}{offset: 40}} \
}
```

MORE EXAMPLES

For more example, please start GUI and invoke the following command. **gui_show_task_assistant -task "Design Planning:PG Planning->Examples->Overview"**.

SEE ALSO

remove_pg_regions(2)
report_pg_regions(2)
report_pg_strategies(2)
set_pg_strategy(2)

create_pg_ring_pattern

Creates a power ground (PG) ring pattern. The pattern created by this command is associated with a region or area of the design by using the **set_pg_strategy** command. The actual ring is inserted into the design by the **compile_pg** command.

SYNTAX

status **create_pg_ring_pattern**
pattern_name
[-nets *net_list*]
[-horizontal_width *width_list*]
[-vertical_width *width_list*]
[-horizontal_spacing *spacing_list*]
[-vertical_spacing *spacing_list*]
[-horizontal_layer *layer*]
[-vertical_layer *layer*]
[-side_width *width_spec*]
[-side_spacing *spacing_spec*]
[-side_layer *layer_spec*]
[-corner_bridge true | false | @*param*}]
[-track_alignment track | half_track |@*var*]
[-parameters *var_list*]
[-via_rule *via_rule_spec*]

Data Types

pattern_name string
net_list list
width_list list
spacing_list list
layer string
width_spec specification
spacing_spec specification
layer_spec specification
param string
var_list list
via_rule_spec specification

ARGUMENTS

pattern_name

Specifies the name of the ring pattern.

-nets *net_list*

Specifies power and ground net names used in this ring pattern. The net names are symbolic names and are mapped to actual net names by the **set_pg_strategy** command.

The net name mapping is by position, where the first net name in the *net_list* list corresponds to the first net name specified by the **set_pg_strategy** command, and so on. This option is required when you specify a net-based via rule, as each net can have a different via requirement. Otherwise, this option is not required.

-horizontal_width *width_list*

Specifies the horizontal ring width. *width_list* specifies the list of horizontal widths, where the first width in the list is the width of the first net, and so on. The unit is um. Each value can also be specified by evaluating the variable in the **-parameters** option. The at character (@) is used as a prefix symbol to specify a parameter and directs the command to replace the value with the value specified by the **set_pg_strategy** command. If this option is not specified, the minimum width defined in the technology file is used for the horizontal width.

-vertical_width *width_list*

Specifies the vertical ring width. *width_list* specifies the list of vertical widths, where the first width in the list is the width of the first net, and so on. The unit is um. Each value can also be specified by evaluating the variable in **-parameters** option. The at character (@) is used as a prefix symbol to specify a parameter. If this option is not specified, the minimum width defined in the technology file is used for the vertical width.

-horizontal_spacing *spacing_list*

Specifies the spacing between horizontal ring segments. *spacing_list* specifies the list of horizontal spacing values, where the first value in the list is the spacing of the first net, and so on. The unit is um. Each value can also be specified by evaluating the variable in **-parameters** option. The at character (@) is used as a prefix symbol to specify a parameter. If this option is not specified, the minimum spacing defined in the technology file is used for horizontal spacing.

-vertical_spacing *spacing_list*

Specifies the spacing between vertical ring segments. *spacing_list* specifies the list of vertical spacing values. The first value in the list is the spacing of the first net, and so on. The unit is um. Each value can also be specified by evaluating the variable in **-parameters** option. The at character (@) is used as a prefix symbol to specify a parameter. If this option is not specified, the minimum spacing defined in the technology file is used for vertical spacing.

-horizontal_layer *layer*

Specifies the horizontal layer name. The layer name can also be specified by evaluating the variable in **-parameters** option. The at character (@) is used as a prefix symbol to specify a parameter. If this option is not specified, the topmost metal layers in the preferred routing direction defined in the technology file is used.

-vertical_layer *layer*

Specifies the horizontal layer name. The layer name can also be specified by evaluating the variable in **-parameters** option. The at character (@) is used as a prefix symbol to specify a parameter. If this option is not specified, the topmost metal layers in the preferred routing direction defined in the technology file is used.

-side_width *width_spec*

Specifies the ring segment width per side. The *width_spec* is defined as follows:

{{**side** : *id_list*}
{**width** : *width_list*}}

where **side** is the keyword to create a side specification. Side contains a list of edge IDs. Each ring edge is indexed by a unique number. Edge numbering starts from 1 at the leftmost vertical edge. If there is more than one leftmost edge, the bottommost edge is the starting edge. The edge number increases by 1 as you proceed around the shape in the clockwise direction.

width is the keyword to create a ring width specification for that side. *width_list* specifies the width list for each net. The unit is um. Each value can also be specified by evaluating the variable in **-parameters** option. The at character (@) is used as a prefix

symbol to specify a parameter.

You can create multiple specifications, where each spec is contained within a brace pair.

The side width specified in this option has a higher priority than the horizontal or vertical width.

-side_spacing *spacing_spec*

Specifies the ring segment spacing per side. The *spacing_spec* is defined as follows:

{{**side** : *id_list*}
{**spacing**: *spacing_list*}}

where **side** is the keyword for side. Side contains a list of edge IDs. Each ring edge is indexed by a unique number. Edge numbering starts from 1 at the leftmost vertical edge. If there are more than one leftmost edges, the bottommost edge is the starting edge. The edge number increases by 1 as you proceed around the shape in the clockwise direction.

spacing is the keyword for ring spacing of that side. *spacing_list* specifies the spacing list between nets. The unit is defined in the technology file. Each value can also be specified by evaluating the variable in **-parameters** option. The at character (@) is used as a prefix symbol to specify a parameter.

You can create multiple specifications, where each spec is contained within a brace pair.

The side spacing specified in this option has a higher priority than the horizontal or vertical spacing.

-side_layer *layer_spec*

Specifies the ring segment layer per side. The *layer_spec* is defined as follows:

{{**side** : *id_list*}
{**layer**: *layer_name*}}

where **side** is the keyword for side. Side contains a list of edge IDs. Each ring edge is indexed by a unique number. Edge numbering starts from 1 at the leftmost vertical edge. If there are more than one leftmost edges, the bottommost edge is the starting edge. The edge number increases by 1 as you proceed around the shape in the clockwise direction.

layer is the keyword for ring layer of that side. *layer_name* specifies the layer name for each net. Each value can also be specified by evaluating the variable in **-parameters** option. The at character (@) is used as a prefix symbol to specify a parameter.

You can create multiple specifications, where each spec is contained within a brace pair.

The side layer specified in this option has a higher priority than the horizontal or vertical layer.

-corner_bridge true | false | @*param*

Specifies whether the pattern creates a corner bridge. You can specify the setting by evaluating the variable *param* in the **-parameters** option. The at character (@) is used as a prefix symbol to specify a parameter. The "true" setting creates bridge connection at all ring corners and connects inner and outer rings of the same net. If this option is not specified, no bridge connection is created for the ring pattern.

-track_alignment track | half_track |@*var*

Specifies the track alignment option of this wire pattern. It can be specified by track or half_track. It can also be specified by evaluating the *var* variable in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable.

-parameters *var_list*

Specifies the list of parameters for this pattern. Each parameter can be evaluated and used as the argument for other options in this command. This allows the pattern to be programmable and reused by passing different parameters in different situations. Each parameter *var* in the list *var_list* is a string. Parameter strings in *var_list* for the **-parameters** option do not contain the at character (@), but the at character is used as a prefix symbol for parameters when used with the other command options.

Parameter values are specified by the **set_pg_strategy** command. See the example section for more details.

-via_rule ***via_rule_spec***

Specifies the via rule for creating vias between horizontal and vertical ring segments. The via rule defines the intersection locations and the via at the locations.

There are several ways to define the intersection locations in the *via_rule_spec* specification. The syntax is defined as follows:

{**intersection** : all} {*via_def*} |
{**intersection** : adjacent} {*via_def*} |
list_of_filter_rules |

The first way is "{**intersection** : all}", where **intersection** is the keyword and **all** refers to all intersections between any ring segments in different directions and different layers.

The second way is "{**intersection** : adjacent}", where **intersection** is the keyword, and **adjacent** refers to all intersections between any two ring segments in different directions only in adjacent layers.

The third way is to define the intersections based on two sets of shapes using *list_of_filter_rules*, which contains a list of filtering rules. Each filtering rule is specified inside a brace pair. The syntax of each filtering rule is as follows:

{{*set_1*} {*filter_1*}} {{*set_2*} {*filter_2*}} {*via_def*} |
{**intersection** : undefined} {*via_def*}

where the two sets are the set of ring segments. Each set can be further filtered to obtain a subset of selected shapes. *set_1* and *set_2* are defined as follows:

{**side** : *id_list*}

where **side** is the keyword for ring side, which contains a list of edge IDs *id_list*.

The filtering operations for ring segments are defined as follows:

{**nets** : *net_list*}
{**layers** : *layer_list*}
{**width** : {*lower_w upper_w*}}

nets, **layers**, **width** keywords can be used to specify the wire net names by *net_list*, wire layer names by *layer_list* or width range by {*lower_w upper_w*}. Ring segments which satisfy the above criteria form a subset of shapes.

For those intersections which do not belong to any filtering rule, the via definition can be specified by

{**intersection** : undefined}{*via_def*}

where **intersection** is the keyword, **undefined** refers to the remaining intersections which do not belong to any filtering rules. By default, vias are not created at the intersections if they do not belong to a filtering rule, unless otherwise specified. Multiple via filtering rules, such as intersection location and via definition, can be defined and separated by braces, where each filter rule is in one brace pair.

With the intersection definition, the via specification {*via_def*} is in the following format:

{**via_master** : *via_master_list* | *via_rule_list*}

where **via_master** is the keyword for both via masters and via master rules. *via_master_list* specifies the list of via masters. Via masters include via contact code defined in the technology file or user-defined via cells. *via_rule_list* is a list of PG via rules. Via master rules are defined by the **set_pg_via_master_rule** command. **NIL** can be used as a keyword to indicate that no via is created. **default** can be used to apply the default vias in the specified location. For each specified intersection, vias are created based on the specified via masters or via master rules. If **NIL** is not specified in the list, the default contact code is used to complete a stacked via or to replace a via when a DRC error occurs.

By default, the default via defined in the technology file is created at each intersection of orthogonal ring segments, where the ring segments are on different layers.

DESCRIPTION

Creates a power ground (PG) ring pattern. The pattern created by this command is associated with a region or area of the design by using the **set_pg_strategy** command, and is inserted into the design by the **compile_pg** command.

The ring pattern specifies the horizontal and vertical ring width values, layer names, spacing values and corner bridge options. The values can be either fixed values or set by using parameters. Via rules between horizontal and vertical ring segments can also be specified.

The ring contour shape, offset to the ring contour and vias between this ring pattern and any other shapes can be specified by **set_pg_strategy** command when associating this pattern to a routing area in the design.

EXAMPLES

The following example creates a ring pattern ring1 for layers M3 and M4. The spacing values are 1 and 2, the width is 5 for horizontal and vertical segments except for side 1 with width 6, the corner bridge option is defined by parameter @flag, and the via master rule via34_2x2, i.e. between layer M3 and M4 with array size 2 by 2.

```
prompt> create_pg_ring_pattern ring1 \
  -parameters {flag} \
  -horizontal_layer M3 -vertical_layer M4 \
  -horizontal_width 5 -vertical_width 5 \
  -horizontal_spacing 1 -vertical_spacing 2 \
  -side_width {{side: 1} {width: 6}} \
  -corner_bridge @flag \
  -via_rule {{intersection: all} {via_master: VIA34}}
```

The following example creates a ring pattern ring2, where the width list {1 2 3} is applied to sides {1 3} and width list {3 4 5} is applied to sides {2 4}. For sides 1 and 3, the innermost ring width is 1, middle ring width is 2 and outermost ring width is 3. For sides 2 and 4, the innermost ring width is 3, middle ring width is 4 and outermost ring width is 5.

```
prompt> create_pg_ring_pattern ring2 \
  -side_width { \
    {{side: 1 3} {width: 1 2 3}} \
    {{side: 2 4} {width: 3 4 5}} \
  }
```

The following example creates a ring pattern ring3, where the width list {1 2 3} is applied to sides {1 2 3 4}. For sides 1, 2, 3 and 4, the innermost ring width is 1, the middle ring width is 2 and the outermost ring width is 3.

```
prompt> create_pg_ring_pattern ring3 \
  -side_width { \
    {side: 1 2 3 4} {width: 1 2 3} \
  }
```

MORE EXAMPLES

For more example, please start GUI and invoke the following command. **gui_show_task_assistant -task "Design Planning:PG Planning->Examples->Overview"**.

SEE ALSO

compile_pg(2)
create_pg_composite_pattern(2)
create_pg_macro_conn_pattern(2)
create_pg_std_cell_conn_pattern(2)
create_pg_wire_pattern(2)
remove_pg_patterns(2)
report_pg_patterns(2)
report_pg_strategies(2)
set_pg_strategy(2)
set_pg_via_master_rule(2)

create_pg_special_pattern

Creates a power ground special pattern. This pattern can be instantiated by **set_pg_strategy** command to create power ground shapes inside the specified routing area.

SYNTAX

status **create_pg_special_pattern**
pattern_name
[-insert_channel_straps *spec*]
[-insert_terminal_alignment_straps *spec*]
[-terminal_alignment_via_rule *via_rule_spec*]
[-insert_power_switch_alignment_straps *spec*]
[-insert_physical_cell_alignment_straps *spec*]
[-honor_max_stdcell_strap_distance *spec*]
[-parameters *var_list*]

Data Types

pattern_name collection
spec specification
via_rule_spec specification
var_list specification

ARGUMENTS

pattern_name

Specifies the name of the special pattern.

-insert_channel_straps *spec*

Specifies the channel strap insertion option specification. Channel straps are inserted inside channels formed by objects if no existing straps are inside the channels. The *spec* specification is as follows:

{
{**layer** : *layer_name*}
{**direction** : horizontal | vertical}
{**width** : *width_value*}
{**spacing** : *spacing_value*}
{**offset** : *offset*}
{**reference** : *offset_reference*}
{**channel_threshold** : *threshold*}
{**track_alignment** : *track* | *half_track*}
{**channel_between_objects** :
macro | placement_blockage | voltage_area |

macro_with_keepout | region_boundary | block}
{**check_one_layer** : true | false}
}

The **layer** keyword followed by the *layer_name* layer specifies the layer for channel strap insertion; this setting is required. The **direction** keyword followed by horizontal or vertical specifies the channel direction; the default is vertical. The **width** keyword followed by *width_value* specifies the channel strap width; the default is the minimum width. The **spacing** keyword followed by *spacing_value* specifies the spacing between channel straps; the default is the minimum spacing. The **offset** keyword followed by *offset* specifies the offset from **reference** to the center of the first strap. The **reference** keyword followed by *offset_reference* specifies the offset reference. The offset_reference could be either left or center. Left means channel left boundary. Center means the centerline of the channel. The default is left. However, if the offset is 0, the reference will be changed to center. The **channel_threshold** keyword followed by *threshold* specifies the lower width bound for narrow channels. Any channels with width smaller than the threshold are not considered for insertion. The default threshold is 0. The **track_alignment** keyword followed by track or half_track specifies how straps align to the track.

The **channel_between_objects** keyword followed by the object name specifies the objects considered for channels such as macros, hard placement blockages, voltage areas, macros with hard keepout, routing area boundary, and blocks. When **macro_with_keepout** is specified, if a hard keepout is not defined for a certain macro, the macro boundary is used for channel detection. When **voltage_area** is specified, channels between embedded voltage areas and disjoint voltage areas are supported, while channels between partially overlapped voltage areas are not supported. If no object is specified using the **channel_between_objects** keyword, by default the tool considers only macros for channels.

The **check_one_layer** keyword followed by true or false specifies when the tool inserts straps in the channel. If **check_one_layer** is false, channel straps are not inserted if any straps at any layers are inside the channel. If **check_one_layer** is true, channel straps are not inserted only if any straps at the specified layer are inside the channel. By default, **check_one_layer** is false.

-insert_terminal_alignment_straps *spec*

Specifies how terminal alignment straps are inserted. If *spec* is "on", terminal alignment straps will be inserted for all terminals at all layers. By default, the terminal alignment straps are created at all layers. Terminal straps are trimmed and snapped to shapes in different directions when cut by blockages or DRCs. If dangling wires are preferred, you can turn off the trim option. If specific layers are preferred, layers can be specified in the spec. The *spec* for trim is described as follows:

{**trim** : true | false}
{**layers** : layerList}

When **trim** is false, terminal alignment straps are not trimmed. **layers** is the keyword for layers and layerList specifies the list of layers for terminal alignment strap insertion.

-terminal_alignment_via_rule *via_rule_spec*

Specifies the via rule *via_rule_spec* between terminal alignment straps. The via rule defines the intersection locations and the via inserted at the locations. This option can be applied only for terminal alignment pattern.

There are several ways to define the intersection locations in the *via_rule_spec* specification, where the syntax is defined as follows:

{**intersection** : all} {*via_def*} |
{**intersection** : adjacent} {*via_def*} |
list_of_filter_rules |

The first method is "{**intersection** : all}", where **intersection** is the keyword and **all** refers to all intersections between any two orthogonal wires in any different layers.

The second method is "{**intersection** : adjacent}", where **intersection** is the keyword and **adjacent** refers to all intersections between any two orthogonal wires only in adjacent layers.

The third method is to define the intersections based on two sets of wire shapes using *list_of_filter_rules*, which contains a list of filtering rules. Each filtering rule is specified inside a curly brace pair. The syntax of each filtering rule is as follows:

{*filter_1*}{*filter_2*}{*via_def*} |
{**intersection** : undefined}{*via_def*}

where *filter_1* refers to the first filtering operation and *filter_2* refer to the second filtering operation. The intersection is defined between the first set and second set of wires based on the filtering rules. The second filtering rule is optional, and the intersection is defined based on the first set of wires to all other terminal alignment straps.

The filtering operation is defined as follows:

{**layers** : *layer_list*} {**width** : {*lower_w upper_w*}}

where the **layers** keyword followed by *layer_list* specifies the wire layer names. The **width** keyword and {*lower_w upper_w*} setting specifies the lower and upper range of possible wire widths. For those intersections which do not belong to any filtering rule, the via definition can be specified as follows:

{**intersection** : undefined}{*via_def*}

where **intersection** is the keyword and **undefined** refers to the remaining intersections which do not belong to any filtering rules. By default, no via is created at the intersections which do not belong to any filtering rules unless specified. Multiple via filtering rules, such as intersection location and via definition, can be defined and separated by curly braces where each filter rule is in one brace pair.

With the intersection definition, the via specification {*via_def*} is in the following format:

{**via_master** : *via_master_list* | *via_rule_list*}

where **via_master** is the keyword for via masters including via contact code defined in the technology file or user-defined via cells. The PG via rules defined by **set_pg_via_master_rule** command can be specified in the list. *via_master_list* specifies the list of via masters. Via masters are created at the intersection center without replication. If offset or specific array dimension is needed, via rules specified by **set_pg_via_master_rule** command can be used. *via_rule_list* is a list of PG via rules. **NIL** can be used as a keyword indicate that no via would be created. **default** can be used to describe the default vias in the specified location. Multiple via rules can be specified and separated by brace pairs. If **NIL** is not specified in the list, default contact code will be used to complete a stacked via or to replace a via with DRCs when applicable.

By default, the default via defined in the technology file will be created at each intersection of orthogonal terminal alignment straps in different layers.

-insert_power_switch_alignment_straps ***spec***

Specifies the power switch alignment strap insertion option. The alignment straps are inserted to align with power switch primary power input pins. The *spec* is defined as follows:

{
{**lib_cells** : *lib_cell_name_list*}
{**layer** : *layer_name*}
{**width** : *width_value*}
{**direction** : horizontal | vertical}
{**offset** : *offset_value*}
{**pitch** : *pitch_value*}
{**track_alignment** : *track* | *half_track*}
{**number** : *number*}
{**pin_layers** : *layer_name_list*}
{**via_master** : *via_master_list*}
}

The **lib_cells** keyword followed by *lib_cell_name_list* specifies the lib cell name list for power switch cells; this option is required. The **layer** keyword followed by *layer_name* specifies the layer for strap insertion, this option is required. The **width** keyword followed by *width_value* specifies the strap width; this option is optional. By default, the pin width is used. The **direction** option followed by horizontal or vertical specifies the alignment strap direction, the default is vertical. The **offset** keyword followed by *offset_value* specifies the offset between strap center and pin shape center. By default, the offset is 0. The **pitch** keyword followed

by *pitch_value* specifies the pitch between alignment strap center. The **track_alignment** keyword followed by track or half_track specifies how straps align track. The **number** keyword followed by *number* specifies the number of alignment straps for each pin shape. By default, the number is 1. The **pin_layers** keyword followed by *layer_name_list* specifies a list of layer names. Only pin shapes on the specified layers are aligned. By default, pins on all layers are aligned. The **via_master** keyword followed by *via_master_list* specifies the via masters used for via creation between the alignment straps and pins. *via_master_list* can include contact codes or design vias defined in the technology file, or via master rule defined by **set_pg_via_master_rule** command. It can also include **default** or **NIL** keywords. By default, the default contact codes are used for via creation.

-insert_physical_cell_alignment_straps *spec*

Specifies the physical cell alignment strap insertion option. Supported cell types include physical_only, well_tap, and filler. The alignment straps are inserted to align with physical cell pins. The *spec* is defined as follows:

{
{**lib_cells** : *lib_cell_name_list*}
{**layer** : *layer_name*}
{**width** : *width_value*}
{**direction** : horizontal | vertical}
{**offset** : *offset_value*}
{**track_alignment** : *track* | *half_track*}
{**number** : *number*}
{**pin_names** : *pin_name_list*}
{**pin_layers** : *layer_name_list*}
{**via_master** : *via_master_list*}
}

The **lib_cells** keyword followed by *lib_cell_name_list* specifies the lib cell name list for physical cells. The **layer** keyword followed by *layer_name* specifies the layer to use for strap insertion; this setting is required. The **width** keyword followed by *width_value* specifies the strap width. This setting is optional. By default, the tool uses the pin width. The **direction** option followed by horizontal or vertical specifies the alignment strap direction. By default, the direction is vertical. The **offset** keyword followed by *offset_value* specifies the offset between strap center and pin shape center, by default, the offset is 0. The **track_alignment** keyword followed by track or half_track specifies how straps align track. The **number** keyword followed by *number* specifies the number of alignment straps for each pin shape. By default, the number is 1. The **pin_names** keyword followed by *pin_name_list* specifies a list of pin names for alignment. By default all pins are considered for alignment. The **pin_layers** keyword followed by *layer_name_list* specifies a list of layer names and only pin shapes on the specified layers are aligned. By default, pins on all layers are aligned. The **via_master** keyword followed by *via_master_list* specifies the via masters used for via creation between the alignment straps and pins. *via_master_list* can include contact codes or design vias defined in the technology file, or via master rule defined by **set_pg_via_master_rule** command. It can also include **default** or **NIL** keywords. By default, the default contact codes are used for via creation.

-honor_max_stdcell_strap_distance *spec*

Specifies the extra strap insertion to honor maximum standard cell rail tail distance option. Extra vertical straps are inserted for rail tails longer than the specified maximum distance. The *spec* is defined as follows:

{
{**layer** : *layer_name*}
{**width** : *width_value*}
{**max_distance** : *threshold*}
{**offset** : *offset_list*}
{**check_layers** : *layer_name*}
}

The **layer** keyword followed by *layer_name* specifies the layer for strap insertion; this setting is required. The **max_distance** keyword followed by *threshold* specifies the maximum length of standard cell rail tails; this setting is required. The **width** keyword followed by *width_value* specifies the strap width. This setting is optional; by default, the tool uses the minimum width for the layer. The **check_layers** keyword followed by *layer_name* specifies target layers for standard cell rail tails distance calculation. User can spefify one or multiple layers. This is an optional setting. By default, it will check all layers.

The **offset** keyword followed by *offset_list* specifies a list of offset values. This setting is optional. By default, extra straps are inserted such that the distance between the inserted strap center and the nearest strap center is *threshold*. If *offset_list* is specified, the largest value from *offset_list* is used to calculate the extra strap distance such that the extra strap is within the rail tail range.

-parameters *var_list*

Specifies the list of parameters for this pattern. Each parameter can be evaluated and used as the argument for other options in this command. This allows the pattern to be programmable and reused by passing different parameters in different situations. Each parameter *var* in the list *var_list* is a string. Parameter strings in *var_list* for the **-parameters** option do not contain the at character (@), but the at character is used as a prefix symbol for parameters when used with the other command options. Parameter values are specified by the **set_pg_strategy** command.

DESCRIPTION

Creates a power ground special pattern. This pattern can be instantiated by the **set_pg_strategy** command to create special power ground shapes inside the specified routing area.

The special pattern can specify channel strap insertion, terminal alignment strap insertion, or extra strap insertion to honor maximum standard cell rail tail distance. Only one type can be specified for one special pattern.

EXAMPLES

The following example creates a special pattern for channel strap insertion. Vertical channels straps are inserted on layer M8 with width 1.0um. Macros, voltage areas and placement blockages are considered for channels.

```
prompt> create_pg_special_pattern channelPattern \
  -insert_channel_straps { \
    {layer: M7}{direction: vertical}{width: 1} \
    {channel_between_objects: {macro placement_blockage voltage_area}} \
  }
```

The following example creates a special pattern for terminal alignment strap insertion. The vias between adjacent layers are specified in the terminal alignment via rule.

```
prompt> create_pg_special_pattern terminalPattern \
  -insert_terminal_alignment_straps on \
  -terminal_alignment_via_rule { \
    {{layers: M8}{layers: M7}{via_master: VIA78}} \
    {{layers: M7}{layers: M6}{via_master: VIA67}} \
  }
```

The following example creates a special pattern for power switch cell pin alignment strap insertion. The alignment straps are inserted on layer M4 for power switches with lib cell name HEAD16DM.

```
prompt> create_pg_special_pattern pwrSwitchAlignmentPattern \
  -insert_power_switch_alignment_straps { \
    {lib_cells: HEAD16DM} {layer: M4} \
  }
```

The following example creates a special pattern for physical cell pin alignment strap insertion. The alignment straps are inserted on

layer M4 for tap cells with lib cell names TAP_REF1 and TAP_REF2.

```
prompt> create_pg_special_pattern \
   phyCellAlignmentPattern \
   -insert_physical_cell_alignment_straps { \
        {lib_cells: TAP_REF1 TAP_REF2} {layer: M4} \
   }
```

The following example creates a special pattern for extra strap insertion to honor maximum standard cell rail tail distance. The tool inserts extra straps of width 2um on layer M8, the rail tail maximum length is 20um, the location is decided by offset value list {20 10}.

```
prompt> create_pg_special_pattern \
   extraStrapPattern \
   -honor_max_stdcell_strap_distance { \
        {layer: M8}{max_distance: 20}{width: 2}{offset: {20 10}} \
   }
```

In addition to the above examples, see the Examples page in the PG Planning section of the Task Assistant for more information. To view the Examples page, start the GUI and invoke the following command: **gui_show_task_assistant -task "Design Planning:PG Planning->Examples->Overview"**.

SEE ALSO

compile_pg(2)
remove_pg_patterns(2)
report_pg_patterns(2)
report_pg_strategies(2)
set_pg_strategy(2)
set_pg_via_master_rule(2)

create_pg_stapling_vias

Creates PG stapling vias for the specified PG nets between parallel PG straps.

SYNTAX

status **create_pg_stapling_vias**
-nets *netname_list*
-from_shapes *shape_collection*
-from_layer *layer*
-to_shapes *shape_collection*
-to_layer *layer*
[-align_track top_layer|bottom_layer]
[-contact_code *contact_code_list*]
[-via_masters *via_rule_list*]
[-ignore_drc]
[-mask mask_one | mask_two | auto]
[-mark_as strap | ring | std_conn | macro_conn]
[-max_array_size integer]
[-no_effect_of_pg_nets_on_each_other]
[-offset *coordinate*]
[-pitch *coordinate*]
[-regions *bbox | bbox_list*]
[-tag *tag_name*]
[-create_via_matrix]

Data Types

netname_list list
shape_collection collection
layer string
contact_code_list list
via_rule_list list
coordinate {float, float}
bbox list
bbox_list list
tag_name string

ARGUMENTS

-nets *net_list*

Specifies a list of PG nets on which to create vias. This is a required option.

-from_shapes *shape_collection*

Specifies the collection of PG shapes to create stapling vias. This option is required.

-from_layer *layer*

Specifies the layer name for PG shapes. This option is required. from_layer and to_layer must be adjacent metal layers.

-to_shapes *shape_collection*

Specifies the collection of PG shapes to create stapling vias. This option is required.

-to_layer *layer*

Specifies the layer name for PG shapes. This option is required. from_layer and to_layer must be adjacent metal layers.

-align_track top_layer | bottom_layer

Specifies the track alignment option for stapling via creation. The argument must be either **top_layer** or **bottom_layer**. If **top_layer** is specified, stapling cuts will be aligned to the routing tracks of the top enclosure layer. If **bottom_layer** is specified, stapling cuts will be aligned to the routing tracks of the bottom enclosure layer. By default, no track alignment is applied.

-contact_code *contact_code_list*

Specifies a list of contact codes to be used during via creation. Contact code names are defined in the technology file. If not specified, the default contact code is used. The option is mutually exclusive with -via_masters.

-via_masters *via_rule_list*

Specifies a list of PG via master rules to be used during via creation. PG via master rules are defined using set_pg_via_master_rule command. The option is mutually exclusive with -contact_code.

-ignore_drc

Specifies the DRC option for stapling via creation. If specified, no drc check is applied for via creation. By default, DRC is checked and fixed for all stapling vias.

-no_effect_of_pg_nets_on_each_other

Specifies if stapling vias of one PG net may cause DRC of another PG net in a list on adjacent rows. By default the tool assumes that one net may affect another, and created stapling vias are loaded into DRC checking engine. This may impact runtime, and if the user is sure that stapling vias with their medtal enclosures have absolutely no effect on adjacent rows, this option will help with runtime.

-mask mask_one | mask_two | auto

Specifies the mask option for stapling via creation. The argument must be one of: **mask_one**, **mask_two**, **auto**. If **mask_one** is specified, mask of all cuts will be assigned as mask_one. If **mask_two** is specified, mask of all cuts will be assigned as mask_two. If **auto** is specified, the tool will automatically assign either mask_one or mask_two to cuts to achieve best stapling rate. By default, no mask is assigned to cuts.

-mark_as strap | ring | std_conn | macro_conn

Specifies the type of the stapling vias to be created. The argument must be one of: **strap**, **ring**, **std_conn** and **macro_conn**. These types refer to strap, ring, standard cell connection and macro connection respectively. By default, **std_conn** is used.

-max_array_size integer

Specifies the maximum via array size for stapling vias. For example, if -max_array_size is specified as 20, then all stapling via arrays created will be 1x20 or smaller. By default, array size is not limited.

-offset coordinate

Specifies the offset in both x and y direction. For stapling vias in horizontal direction, x offset means shrink the intersection area

by specified amount; y offset means shift the vias in y direction by certain amount. For example, for horizontal direction stapling, -offset {1.0, 0.1} means that in y axis, the cuts are moved up by 0.1um; while in x axis, all the intersections will be shrank by 1.0um from both ends.

-pitch coordinate

Specifies the stapling via pitch in both x and y direction. By default, minimum cut spacing will be used as spacing between cuts. Only horizontal pitch is honored.

-regions *bbox* | *bbox_list*

Specifies a bounding box or a list of bounding boxes within which to create PG vias. The vias are created at the power strap intersections within this bounding box. If not specified, the design boundary is used as the bounding box.

-tag *tag_name*

Specifies a tag name to assign to all new vias created by this command. This tag name is used as contents of an user attribute tag for filtering collections at later stages (see the example in the EXAMPLES section). By default, no tags are assigned.

-create_via_matrix

If specified, the stapling vias will be stored as via matrixes.

DESCRIPTION

This command is used for stapling via creation between parallel straps on adjacent as well as non-adjacent layers, usually between standard cell rails on M1/M2. Only stapling vias in the horizontal direction are supported. By default the command will check DRC against std cells and signal/clock nets. All created vias will be assigned attribute is_via_staple true. It does not honor app options plan.pgrouter.honor_signal_route_drc and plan.pgroute.honor_std_cell_drc.

EXAMPLES

The following example creates PG stapling vias for net VDD between M1/M2 horizontal shapes. The mask of cuts is automatically assigned, and the maximum array size is 20.

```
prompt> create_pg_stapling_vias \
   -nets VDD \
   -from_layer M1 \
   -to_layer M2 \
   -from_shapes [get_shapes -of_objects VDD -filter "layer_name == M1"] \
   -to_shapes [get_shapes -of_objects VDD -filter "layer_name == M2"] \
   -mask auto \
   -max_array_size 20
```

The following example creates collections of vias that are tagged with the name pg1.

```
prompt> create_pg_stapling_vias \
   -nets VDD \
   -from_layer M1 \
   -to_layer M2 \
   -from_shapes [get_shapes -of_objects VDD -filter "layer_name == M1"] \
   -to_shapes [get_shapes -of_objects VDD -filter "layer_name == M2"] \
```

```
    -mask auto \
    -max_array_size 20 \
    -tag pg1
prompt> set vias [get_vias -filter tag==pg1]
```

The following example creates PG stapling vias for net VDD between non-adjacent M0/M2 horizontal shapes. It then checks number of vias with attribute is_via_staple==true

```
prompt> create_pg_stapling_vias \
    -nets VDD \
    -from_layer M0 \
    -to_layer M2 \
    -from_shapes [get_shapes -of_objects VDD -filter "layer_name == M0"] \
    -to_shapes [get_shapes -of_objects VDD -filter "layer_name == M2"]
prompt> sizeof_collection [get_vias -filter is_via_staple==true]
```

MORE EXAMPLES

For more example, please start GUI and invoke the following command. **gui_show_task -task "Design Planning:PG Planning->Examples->Overview"**

SEE ALSO

compile_pg(2)
create_pg_vias(2)

create_pg_std_cell_conn_pattern

Creates a standard cell rail connection pattern. This pattern can be instantiated in a design by **set_pg_strategy** command to create standard cell rails.

SYNTAX

status **create_pg_std_cell_conn_pattern**
pattern_name
[-rail_width {*top_width* | *@var1 bottom_width* | *@var2*}]
[-rail_shift {*top_shift* | *@var1 bottom_shift* | *@var2*}]
[-rail_mask {*top_mask* | *@var1 bottom_mask* | *@var2*}]
[-layers *layers* | *@var*]
[-check_std_cell_drc true | false | *@var*]
[-mark_as_follow_pin true | false | *@var*]
[-parameters *var_list*]

Data Types

pattern_name string
top_width float
var1 string
bottom_width float
var2 string
top_shift string
bottom_shift string
layers list
var string
var_list list
top_mask string (mask_one|mask_two|mask_three|follow_pin)
bottom_mask string (mask_one|mask_two|mask_three|follow_pin)

ARGUMENTS

pattern_name

Specifies the name of the standard cell rail connection pattern.

-rail_width {*top_width* | *@var1 bottom_width* | *@var2*}

Specifies the top cell row rail width and bottom cell row width. This option is optional. By default, the tool uses the longest PG pin width. The units are defined in the technology file. Each width can also be specified by evaluating the *var1* and *var2* variables in **-parameters** option. The at character (@) is used as a prefix to evaluate the variable. If only one width is specified, it is applied to both top and bottom width values. If rail width is not specified, the standard cell pin width is used by default. This option is required when a design does not contain any standard cells.

-rail_shift {*top_shift* | *@var1 bottom_shift* | *@var2*}

Specifies the shift distance of rails. The units are defined in the technology file. The rail shift can also be specified by evaluating the *var1* and *var2* variables in the **-parameters** option. The at character (@) is used as a prefix to evaluate the variable. A positive distance shifts the bottom and top rails into the cell rows. A negative distance shifts the bottom and top rails out from the cell rows. If only one shift value is specified, it is applied to both top and bottom shift values. The shift value is applied to the center of rail with respect to the row boundary, that is, a shift value of 0 can align the center of rail with the row boundary. By default, the rails are aligned with standard cell pins.

-rail_mask {*top_mask* | *@var1 bottom_mask* | *@var2*}

Specifies the top cell row rail mask and bottom cell row mask. This option is optional. By default, the rail is not colored with any mask. Each mask can also be specified by evaluating the *var1* and *var2* variables in **-parameters** option. The at character (@) is used as a prefix to evaluate the variable. If only one mask is specified, it is applied to both top and bottom masks. The valid keyword for rail mask is one of mask_one, mask_two, mask_three and follow_pin. "follow_pin" refers to assigning same mask of std cell pins to std rails. Report command **report_pg_patterns** reports the rail mask in the same order, i.e. top and bottom masks.

-layers *layers* | *@var*

Specifies the list of layer names for standard cell connection. By default, the standard cell power ground pin layer is used for connection. The list of layers can be specified by evaluating the variable *@var*. The variable *var* is specified in **-parameters** option. The at character (@) is used as a prefix keyword to evaluate the variable. If layers are not specified, the standard cell pin layer is used for rail creation. This option is required when a design does not contain any standard cells.

-check_std_cell_drc true | false | *@var*

Specifies the option to check DRCs between the rail and the standard cells. For example, if standard cells are not placed, the DRCs between standard cell pins and PG rails can be ignored. This option can be set to "true", "false" or by evaluating the *var* variable in the **-parameters** option. The at character (@) is used as a prefix keyword to evaluate the variable. By default, the option is "false", and DRCs between rails and standard cells are ignored for faster runtime.

-mark_as_follow_pin true | false | *@var*

Specifies the output shape_use of the generated rails. If set the option false, the generated rail will have "lib_cell_pin_connect" attribute in shape_use. If it is true, it will be "follow_pin". The at character (@) is used as a prefix keyword to evaluate the variable. By default, the option is "false".

-parameters *var_list*

Specifies the list of parameters. Each parameter can be evaluated and used by other options in this command. This allows the pattern to be programmable and reused by passing different parameters in different situations. Each parameter *var* in the list *var_list* is a string and the at character (@) is used as a prefix to evaluate the parameter, such as *@var*. See the example section for more details.

DESCRIPTION

Creates a standard cell connection pattern. This pattern can be instantiated in a design by the **set_pg_strategy** command to create standard cell connections. For standard cell rails, macro and pad pins are not connected and floating segments are always kept.

EXAMPLES

The following example creates a standard cell connection pattern std_pattern_1 at layer M1.

```
prompt> create_pg_std_cell_conn_pattern \
  std_pattern_1 \
  -layers {M1}
```

The following example creates a standard cell connection pattern named std_pattern_2 at a programmable layer by using the parameter **my_metal2**. The rail width and shift distance are also specified by parameters, **w** and **dist**, respectively. When referring to this pattern with the **set_pg_strategy** command, you can program the actual layer name for the metal layer, for example "Metal2" or "M2". Parameters allow this pattern to be reused by designs with different technologies.

```
prompt> create_pg_std_cell_conn_pattern \
  std_pattern_2 \
  -parameters {my_metal2 w dist} \
  -layers {@my_metal2} \
  -rail_width {@w @w} \
  -rail_shift {@dist}
prompt> set_pg_strategy \
  s1 \
  -core \
  -pattern {{name: std_pattern_2}{nets: VDD}{parameters: {M2 2 10}}} \
  -extension {{stop: 10} {layers: M2}}
```

MORE EXAMPLES

For more example, please start GUI and invoke the following command. **gui_show_task -task "Design Planning:PG Planning->Examples->Overview"**

SEE ALSO

compile_pg(2)
create_pg_composite_pattern(2)
create_pg_macro_conn_pattern(2)
create_pg_ring_pattern(2)
create_pg_wire_pattern(2)
remove_pg_patterns(2)
report_pg_patterns(2)
report_pg_strategies(2)
set_pg_strategy(2)

create_pg_strap

Creates power ground straps. Vias are also created between the new strap shapes and existing shapes.

SYNTAX

status **create_pg_strap**
-layer *layer_name*
-direction horizontal | vertical
-width *value*
-net *net_name*
[-start *value*]
[-stop *value*]
[-pitch *value*]
[-low_end *value*]
[-high_end *value*]
[-extend_low *ext_type*]
[-extend_high *ext_type*]
[-via_rule *via_rule_spec*]
[-drc no_check | check_but_no_fix]
[-show_phantom]
[-tag *tag_name*]
[-mark_as *pg_type*]
[-mask mask_one | mask_two | mask_three]
[-mask_constraint *constraint_name*]

Data Types

layer_name string
value float
net_name string
ext_type specification
via_rule_spec specification
tag_name string
pg_type specification
constraint_name string

ARGUMENTS

-layer *layer_name*

Specifies the layer name on which to create the PG strap. This is a required option.

-direction horizontal | vertical

Specifies the direction of the PG strap. This is a required option. Legal directions are "horizontal" and "vertical".

-width *value*

Specifies the width of the PG strap. This is a required option. *value* is a float value with um as the unit.

-net *net_name*

Specifies net name of the PG strap by the string *net_name*. This is a required option.

-start *value*

Specifies the starting position of the PG strap. This is a required option. *value* is a float value with um as the unit.

-stop *value*

Specifies the stopping point of the PG strap. This option works with **-pitch** option to create multiple PG straps.

-pitch *value*

Specifies the pitch of the PG straps. This option works with **-stop** option to create multiple PG straps.

-low_end *value*

Specifies the low end coordinate of the PG strap. It could be specified as a fixed *value* with um as the unit. This option is not required. By default, the strap is extended to the core area boundary.

-high_end *value*

Specifies the high end coordinate of the PG strap. It could be specified as a fixed *value* with um as the unit. This option is not required. By default, the strap is extended to the core area boundary.

-extend_low *ext_type*

Specifies the extension type for the low end of PG strap. *ext_type* should be one of **core**, **design_boundary**, **design_boundary_and_generate_pin**, **first_target**, **innermost_ring**, **outermost_ring** or **pad_ring**.

If **-low_end** is not specified, by default the low end of PG strap is extended to the core area boundary. If **-low_end** is specified, by default the low end of PG strap stops at the value specified by **-low_end** and no extension is performed. If both **-extend_low** and **-low_end** are specified, the extension is perform based on the value specified by **-low_end**.

-extend_high *ext_type*

Specifies the extension type for the high end of PG strap. *ext_type* should be one of **core**, **design_boundary**, **design_boundary_and_generate_pin**, **first_target**, **innermost_ring**, **outermost_ring** or **pad_ring**.

If **-high_end** is not specified, by default the high end of PG strap is extended to the core area boundary. If **-high_end** is specified, by default the high end of PG strap stops at the value specified by **-high_end** and no extension is performed. If both **-extend_high** and **-high_end** is specified, the extension is perform based on the value specified by **-high_end**.

-via_rule *via_rule_spec*

Specifies the via rule *via_rule_spec* between the new PG strap and existing shapes including existing PG shapes, macro pins and terminals (top-level block pins). The via rule defines the intersection locations and the via at the locations.

There are three methods to define the intersection locations in the specification *via_rule_spec*, where the syntax is defined as follows:

{**intersection** : all} {*via_def*} |
{**intersection** : adjacent} {*via_def*} |
list_of_filter_rules

The first method is "{**intersection** : all}", where **intersection** is the keyword, and **all** refers to all intersections between the PG strap and any orthogonal existing shapes in any different layers. Existing shapes include PG shapes such as straps, rings, standard cell rails and macro connections, as well as macro pins and terminals.

The second method is "{**intersection** : adjacent}", where **intersection** is the keyword, and **adjacent** refers to all intersections between the PG strap and any orthogonal existing shapes only in adjacent layers.

The third method is to define the intersections based on the PG strap and a set of shapes using *list_of_filter_rules*, which contains a list of filtering rule. Each filtering rule is inside a curly brace pair. The syntax of each filtering rule is as below:

{{*set*}{*filter*}}{*via_def*} |
{**intersection** : undefined}{*via_def*}

where *set* refers to the set of shapes and *filter* refers to the filtering operations for this set. *set* is defined as follows:

{**existing** : *shape_type*} |
{**macro_pins** | **terminals** : all | *pin_names*}

where **existing** is the keyword for existing PG wires, **macro_pins** is the keyword for macro pins and **terminals** is the keyword for terminals. Existing wires are specified by shape type *shape_type*, which could be ring, strap, macro_conn and std_conn.

The keywords ring, strap, std_conn, and macro_conn specify ring, strap, standard cell, and macro pin connection types. Macro pins or terminals could be specified by the all keyword, or by a list of pin names in *pin_names*. The filtering operations of the set of existing shapes are defined as follows:

{**nets** : *net_list*}
{**layers** : *layer_list*}
{**width** : {*lower_w upper_w*}}

nets, **layers**, **width** keywords can be used to specify the net names by *net_list*, wire layer names by *layer_list* or width range by {*lower_w upper_w*}.

By specifying the set of shapes certain filtering operations by net names, layer names, wire width ranges or types, the via locations are the intersections between the subset of shapes and the PG strap. For those intersections which do not belong to any filtering rule, the via definition can be specified by

{**intersection** : undefined}{*via_def*}

where **intersection** is the keyword, **undefined** refers to the remaining intersections which do not belong to any filtering rules. By default, vias are created only if the intersection matches one or more filtering rules. Multiple via filtering rules, that is, intersection location and via definition, can be defined and separated by curly braces where each filter rule is in one curly brace pair.

With the intersection definition, the via specification {*via_def*} uses the following format:

{**via_master** : *via_master_list* | *via_rule_list*}

where **via_master** is the keyword for via masters including via contact code defined in the technology file or user-defined via cells, as well as the PG via rules defined by **set_pg_via_master_rule** command. The *via_master_list* specifies the list of via masters. Via masters are created at the intersection center without replication. If offset or multiplication is needed, via rules specified by **set_pg_via_master_rule** command can be used. *via_rule_list* is a list of PG via rules. **NIL** can be used as a keyword indicate that no via is created. **default** can be used to describe the default vias in the specified location. If **NIL** is not specified in the list, the default contact code is used to complete a stacked via or to replace a via with DRCs when applicable.

By default, the default via defined in the technology file is created at each intersection between the PG strap and any orthogonal shapes in different layers.

-drc no_check | check_but_no_fix

Specifies the DRC option for PG strap creation. The argument following **-drc** must be either no_check or check_but_no_fix. no_check means that no DRC check is performed. check_but_no_fix means that DRC check is performed, DRC errors are reported, but no DRC fixing is performed. The PG strap and associated vias are created, based on the specification when this

option is used. By default, DRC is checked and fixed for the PG strap.

-show_phantom

Generates report for wires/vias that should be created but failed due to fixing DRC violation. Detailed information will be displayed in error browser. The report includes two parts, phantom wire and phantom via. Wire information and DRC violation are included in one phantom wire. Only DRC violations that cut part of wire shape while fixing will be included in phantom wire. Abutted phantom wires will be merged and reported together. Via information and DRC violation are included in one phantom via. Via information includes top/bottom wire information, via def, net name. DRC violation shows initial violation before fixing. If **-drc** no_check or check_but_no_fix is specified, phantom via will not be generated.

-tag *tag_name*

Specifies a tag name to assign to all new vias and shapes created by this command. This tag name is used as contents of an user attribute tag for filtering collections at later stages (see the example in the EXAMPLES section). By default, no tags are assigned.

-mark_as strap | ring | std_conn | macro_conn

Specifies the type of the PG strap and associated vias to be created. The argument must by on of these keywords: strap, ring, std_conn and macro_conn. These types refer to strap, ring, standard cell connection and macro connection respectively. By default, the strap type is used.

-mask mask_one | mask_two | mask_three

Specifies the mask option for the PG strap. By default, the PG strap is not colored with a mask identifier if not specified. The type of mask can be specified by one of the keywords, mask_one, mask_two, mask_three.

-mask_constraint *constraint_name*

Specifies the mask constraint for the PG strap. The PG mask constraint is defined by **set_pg_mask_constraint** command. The PG strap will be checked against the specified constraint. If the constraint is not satisfied, the strap will not be created with error message issued. By default, PG mask constraint is not checked.

DESCRIPTION

Creates one or more power ground straps. Vias are also created between the new strap shape and existing shapes. The command is used to create one or multiple straps in special cases where pattern-based PG cannot create power ground straps.

EXAMPLES

The following example creates a VDD PG strap on layer M3, center at 100um, width as 3um, and horizontal direction. Both ends are extended to core area boundary by default.

```
prompt> create_pg_strap \
  -layer M3 \
  -direction horizontal \
  -width 3 \
  -net VDD \
  -start 100
```

The following example creates VDD PG strap on layer M3, starting at 100um and stopping at 150um with pitch 20um, width as 3um,

and horizontal direction. Both ends are extended to core area boundary by default.

```
prompt> create_pg_strap \
  -layer M3 \
  -direction horizontal \
  -width 3 \
  -net VDD \
  -start 100 \
  -stop 150 \
  -pitch 20
```

The following example creates a VDD PG strap on layer M3, center at 100um, width as 3um, horizontal direction, low end at 50um and high end 300um. DRC is checked but not fixed. All the new shapes are marked as macro connection.

The low end is extended to the innermost ring. The high end is extended to the design boundary with pin generated. Vias are created between the PG strap and rings using a specified rule, but not for macro pins. Default vias are inserted at any remaining intersections.

```
prompt> create_pg_strap \
      -layer M3 \
      -direction horizontal \
      -width 3 \
      -net VDD \
      -start 100 \
      -low_end 50 \
      -high_end 300 \
      -extend_low innermost_ring \
      -extend_high design_boundary_and_generate_pin \
      -drc check_but_no_fix \
      -via_rule { \
            {{existing: ring} {via_master: VIA34}} \
            {{macro_pins: all} {via_master: nil}} \
            {{intersection: undefined} {via_master: default}} \
      }
```

The following example creates a VDD PG strap on layer M3, starts at 100um, ends at 300um, pitch as 10um, width as 3um, and horizontal direction. Both ends are extended to core area boundary by default.

```
prompt> create_pg_strap \
  -layer M3 \
  -direction horizontal \
  -width 3 \
  -net VDD \
  -start 100 \
  -stop  300 \
  -pitch 10 \
```

The following example creates collections of wires and vias that are tagged with the name pg1.

```
prompt> create_pg_strap \
  -layer M3 \
  -direction horizontal \
  -width 3 \
  -net VDD \
  -start 100 \
  -stop  300 \
  -pitch 10 \
```

```
  -tag pg1
prompt> set wires [get_shapes -filter tag==pg1]
prompt> set vias [get_vias -filter tag==pg1]
```

SEE ALSO

compile_pg(2)
create_pg_composite_pattern(2)
create_pg_macro_conn_pattern(2)
create_pg_ring_pattern(2)
create_pg_std_cell_conn_pattern(2)
create_pg_vias(2)
remove_pg_patterns(2)
report_pg_patterns(2)
report_pg_strategies(2)
set_pg_strategy(2)
set_pg_mask_constraint(2)

create_pg_vias

Creates PG vias in the specified bounding box for the specified PG nets.

SYNTAX

status **create_pg_vias**
-nets *netname_list*
[-within_bbox *bboxes*]
[-from_types *type_list*]
[-from_layers *layer_list*]
[-to_types *type_list*]
[-to_layers *layer_list*]
[-via_masters *via_master_list*]
[-allow_parallel_objects]
[-insert_additional_vias]
[-drc no_check | check_but_no_fix]
[-show_phantom]
[-mark_as *usage_type*]
[-tag *tag_name*]
[-shapes *shape_list*]
[-pins *pin_name_list*]
[-pin_collection *pin_collection*]
[-start *coordinate_list*]
[-pitch *coordinate_list*]
[-create_via_matrix]
[-blockage {*blockage_spec*}]
[-create_ml_data]
[-use_ml_model]

Data Types

netname_list list
bboxes list
type_list list
layer_list list
via_master_list list
usage_type string
tag_name string
shape_list list
pin_name_list string
pin_collection collection
coordinate_list list
blockage_spec specification

ARGUMENTS

-nets ***netname_list***

Specifies a list of PG nets on which to create vias. This is a required option.

-within_bbox ***bboxes***

Specifies a bounding box or a list of bounding boxes within which to create PG vias. The vias are created at the power strap intersections within this bounding box. By default, design boundaries are used.

-from_types ***type_list***

Specifies a list of PG object types. The PG types should include one or more of the following keywords: **ring**, **stripe**, **lib_cell_pin_connect**, **macro_pin_connect**,**follow_pin**, **core_wire**, **macro_pin**, **shield_route**, **pwrswitch_pin**, **user_route**, and **terminal**. These types are for PG rings, PG straps, standard cell pin connections, macro pin connections, macro pins, power switch pins, user routes and terminals respectively. This option is not required. By default, all of these types are considered for via creation.

-from_layers ***layer_list***

Specifies the layer names for the PG objects. *layer_list* is a list containing layer names. This option is not required. By default, objects on all layers are considered for via creation.

-to_types ***type_list***

Specifies a list of PG object types. The PG types should include one or more of the following keywords: **ring**, **stripe**, **lib_cell_pin_connect**, **macro_pin_connect**, **follow_pin**, **core_wire**, **macro_pin**, **shield_route**, **pwrswitch_pin**, **user_route**, and **terminal**. These types are for PG rings, PG straps, standard cell pin connections, macro pin connections, macro pins, power switch pins, user routes and terminals respectively. This option is not required. By default, all of these types are considered for via creation.

-to_layers ***layer_list***

Specifies the layer names of PG objects. *layer_list* is a list of layer names. This option is not required. When not specified, objects on all layers are considered for via creation.

-via_masters ***via_master_list***

Specifies the list of via masters to be used during via creation. Contact code names defined in the technology file, custom via definitions, via rule names defined by **set_pg_via_master_rule**, and the **default** keyword can be used in the list. Given one intersection, the feasible via rules are tried for via creation, followed by the feasible specified custom via contact definitions if all via rules do not apply to this intersection, followed by the specified contact codes in technology file, followed by any feasible contact codes or custom via definitions if none of the previous conditions apply to the intersection. If **-via_masters** is not specified, the default contact code is used to complete a stacked via or to replace a via with DRCs when applicable. By default, the first DRC-clean default contact code is used for via insertion for each intersection.

-allow_parallel_objects

Specifies that vias are created between parallel objects. By default, vias are only created between orthogonal objects.

-insert_additional_vias

Specifies that vias are created between intersections with existing vias. By default, additional vias are not created for one intersection where existing vias are present for that intersection.

-drc no_check | **check_but_no_fix**

Specifies the DRC option for PG via creation. The argument following **-drc** must be either no_check or check_but_no_fix. no_check means that no DRC check is performed. check_but_no_fix means that DRC check is performed, DRC errors are reported, but no DRC fixing is performed. By default, DRC is checked and fixed for the PG via.

-show_phantom

Generates report for vias that should be created but failed due to unfixed DRC violation. Detailed information will be displayed in error browser. Via information and DRC violation are included in one phantom via. Via information includes top/bottom wire information, via def, net name. DRC violation shows initial violation before fixing. If **-drc** no_check or check_but_no_fix is specified, phantom via will not be generated.

-mark_as *usage_type*

Specifies the type of the created vias. The argument must be one of the following:

- **stripe**: Power strap
- **ring**: Power ring
- **lib_cell_pin_connect**: Standard cell connection
- **macro_pin_connect**: Macro connection
- **user_route**: User route
- **follow_pin**: Standard cell connection
- **shield_route**: Shield route

If you do not specify this option, the tool uses the shape use of the bottom layer shape for the created vias.

-tag *tag_name*

Specifies a tag name to assign to all new vias and shapes created by this command. This tag name is used as a user attribute for filtering collections at later design stages (see the example in the EXAMPLES section). By default, no tags are assigned.

-shapes *shape_list*

Specifies a list of PG shapes for which to create pg vias. If specified, The command will not search for PG shapes inside specified region, but create pg vias at intersections among the specified shapes.

-pins *pin_name_list*

Specifies a list of PG pin name for which to create pg vias. If specified, The command will not search for PG shapes inside specified region, but create pg vias at intersections among the specified pins. This option is mutual exclusive with option **-pin_collection**.

-pin_collection *pin_collection*

Specifies a collection of PG pins for which to create pg vias. If specified, The command will not search for PG shapes inside specified region, but create pg vias at intersections among the specified pins. This option is mutual exclusive with option **-pins**.

-start coordinate_list

Specifies a list of coordinates as the starting points for array based via insertion flow. The number of coordinates specified should be the same as the number of nets specified by -nets option. This option should be specified together with -pitch option. Note that -via_master option is required for this array based flow.

-pitch coordinate_list

Specifies a list of coordinates as the X/Y pitch for array based via insertion flow. The number of coordiantes specified should be no more than the number of coordinates specified by -start option. The last value in the list will be used if the number of pitches is

less than the number of starts. This option should be specified together with -pitch option. Note that -via_master option is required for this array based flow.

-create_via_matrix

If specified, the command will create via matrixes instead of single vias or via arrays. Currently this option is only honored when -start and -pitch options are specified.

-blockage {*blockage_spec*}

Specifies the routing blockage in the power ground network. The option only applys to start/pitch mode. You can specify multiple blockages within the braces, one blockage per group. Blockages are separated by braces and contain the keywords **nets:**, **layers:**, and a target specification as follows:

{{nets: *nets*} {layers: *layers*} {*target*}

where *target* is a voltage area, block, macro, macro with hard keep-out-margin (KOM), PG region, hard placement blockage, or polygon that is specified as follows:

{voltage_areas : *voltage_areas* | macros : *macro_names* |
macros_with_keepout : *macro_names* | placement_blockages : all |
polygon : {*polygon_area*} | blocks : *blocks* |
pg_regions : *region_list* }

The net names following the *nets* keyword are a subset of the nets specified by **-nets** in *pattern_expr* from **-pattern** option. If you do not specify the **nets** keyword, the blockage applies to all the nets in the strategy. The layer names following the **layers** keyword are the routing layers on which power or ground straps are blocked. If you do not specify the **layers** keyword, the blockage applies to all routing layers. Use one of the **voltage_areas**, **macros**, **polygon**, **blocks** or **pg_regions** keywords to represent the blockage area. *polygon_area* is specified as a list of coordinate points.

-create_ml_data

This option enables this command to create training data for training ML model for fixability prediction. When this option is set, the command only creates ML training data, not creating PG wires/vias.

-use_ml_model

This option enables this command to load pre-trained ML model for fixability prediction.

DESCRIPTION

Creates PG vias in the specified bounding box for the specified PG nets. This command is used for via creation in certain corner cases where the pattern-based PG creation cannot handle.

EXAMPLES

The following example creates PG vias for nets VDD and VSS in bounding box {{900um 900um} {1100um 1100um}}.

```
prompt> create_pg_vias \
   -within_bbox {{900 900} {1100 1100}} \
   -nets {VDD VSS}
```

The following example creates PG vias using 4 threads for nets VDD and VSS in bounding box {{900um 900um} {1100um

1100um}}.

```
prompt> set_host_options -max_cores 4
prompt> create_pg_vias \
   -within_bbox {{900 900} {1100 1100}} \
   -nets {VDD VSS}
```

The following example creates PG vias for nets VDD and VSS in bounding box {{900um 900um} {1100um 1100um}}. Vias are created between rings on layer M5 and standard cell connections on layer M1. Additional vias are allowed and DRC is skipped. All the vias are marked as ring type.

```
prompt> create_pg_vias \
   -within_bbox {{900 900} {1100 1100}} \
   -nets {VDD VSS} \
   -from_types ring \
   -from_layers M5 \
   -to_types lib_cell_pin_connect \
   -to_layers M1 \
   -drc no_check \
   -insert_additional_vias \
   -mark_as ring
```

The following example creates collections of vias and wires that are tagged with the name pg1.

```
prompt> create_pg_vias \
   -within_bbox {{900 900} {1100 1100}} \
   -nets {VDD VSS} -tag pg1
prompt> set vias [get_vias -filter tag==pg1]
prompt> set wires [get_shapes -filter tag==pg1]
```

The following example creates an array of vias for net VDD staring from coordinate {5 5} with X-pitch 10 and Y-pitch 10 using contact code VIA1.

```
prompt> create_pg_vias \
   -within_bbox {{0 0} {1100 1100}} \
   -nets {VDD} \
   -start {{5 5}} \
   -pitch {{10 10}} \
   -via_masters VIA1
```

The following example creates ML training data in memory, output training data to files in the directory ./pg_model, and train ML model based on the training data. The trained ML model is stored in the directory ./pg_model.

```
prompt> create_pg_vias -nets {VDD} -create_ml_data
prompt> train_pg_ml_model
```

The following example creates ML training data in memory, output training data to files in directory pg_data_1, and train ML model based on the training data. The trained ML model is stored in the directory ./pg_model.

```
prompt> create_pg_vias -nets {VDD} -create_ml_data
prompt> create_pg_ml_data -output_directory pg_data_1
prompt> train_pg_ml_model -input_directory {pg_data_1}
```

The following example loads pre-trained ML model in the directory ./pg_model for fixability prediction to reduce runtime of PG via

creation.

```
prompt> create_pg_vias -nets {VDD} -use_ml_model
```

SEE ALSO

compile_pg(2)
create_pg_composite_pattern(2)
create_pg_macro_conn_pattern(2)
create_pg_ring_pattern(2)
create_pg_std_cell_conn_pattern(2)
create_pg_strap(2)
get_shapes(2)
get_vias(2)
remove_pg_patterns(2)
report_pg_patterns(2)
report_pg_strategies(2)
set_host_options(2)
set_pg_strategy(2)
create_pg_ml_data(2)
train_pg_ml_model(2)

create_pg_wire_pattern

Creates a power ground wire pattern. This pattern can be used to create a hierarchical composite pattern with the **create_pg_composite_pattern** command or instantiated in a design with the **set_pg_strategy** command to create a power ground mesh.

SYNTAX

status **create_pg_wire_pattern**
pattern_name
-direction horizontal | vertical | @*var*
-layer *layer_name* | @*var*
[-center *value* | @*var*]
[-low_end_reference_point *value* | @*var*]
[-high_end_reference_point *value* | @*var*]
[-extend_low boundary | first_target | @*var*]
[-extend_high *ext_type* | @*var*]
[-width minimum | *value_list* | @*var*]
[-pitch {*x_value y_value*} | @*var*]
[-spacing interleaving | minimum | *value_list* | @*var*]
[-trim true | false | @*var*]
[-track_alignment track | half_track | @*var*]
[-mask *mask_value*]
[-mask_constraint *constraint_name*]
[-parameters *var_list*]

Data Types

pattern_name string
var string
layer_name string
value float
mask_name string
constraint_name string
ext_type specification
value_list list
x_value float
y_value float
var_list list

ARGUMENTS

pattern_name

Specifies the name of the wire pattern.

-direction horizontal | vertical | @*var*

Specifies the direction of the wire pattern. This is a required option. The direction could be "horizontal", or "vertical", or specified by evaluating the variable *var* in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable.

-layer *layer_name* | @*var*

Specifies the layer name of the wire pattern. This is a required option. The layer can be specified by using *layer_name* or by evaluating the variable *var* in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable.

-center *value* | @*var*

Specifies the center of the first wire in this wire pattern. The center coordinate is relative to the pattern origin (0, 0). *value* is a float value with units defined in the technology file. The center is specified as a fixed *value* or by evaluating the variable *var* in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable. By default, the center is 0,0.

-low_end_reference_point *value* | @*var*

Specifies the low end coordinate. The coordinate is relative to the pattern origin (0,0). The low end can be specified as a fixed *value* or by evaluating the variable *var* in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable. By default, the low end of wire is extended to the left or bottom side of the routing area specified in **set_pg_strategy** command.

-high_end_reference_point *value* | @*var*

Specifies the high end coordinate. The coordinate is relative to the pattern origin (0,0). The high end can be specified as a fixed *value* or specified by evaluating the variable *var* in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable. By default, the high end of wire is extended to the right or top side of the routing area specified in **set_pg_strategy** command.

-extend_low boundary | first_target | @*var*

Specifies the extension type for the low end of wire pattern. **boundary** refers to the left or bottom side of routing area boundary. **first_target** refers to other wires within the routing area. The low extent can also be specified by evaluating the *var* variable in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable. The routing area is specified in **set_pg_strategy** command.

If **-low_end_reference_point** is not specified, by default the low end of wire pattern is extended to routing area boundary. If **-low_end_reference_point** is specified, by default the low end of wire pattern stops at the value specified by **-low_end_reference_point** and no extension is performed. If both **-extend_low** and **-low_end_reference_point** is specified, the **-extend_low** will be ignored.

-extend_high boundary | first_target | @*var*

Specifies the extension type for the high end of wire pattern. **boundary** refers to the right or top side of routing area boundary. **first_target** refers to other wires within the routing area. The high extent can also be specified by evaluating the variable *var* in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable. The routing area is specified in **set_pg_strategy** command.

If **-high_end_reference_point** is not specified, by default the high end of wire pattern is extended to routing area boundary. If **-high_end_reference_point** is specified, by default the high end of wire pattern stops at the value specified by **-high_end_reference_point** and no extension is performed. If both **-extend_high** and **-high_end_reference_point** is specified, the **-extend_high** will be ignored.

-width minimum | *value_list* | @*var*

Specifies the width of the wire pattern. **minimum** refers to the minimum wire width defined in the technology file. *value_list* contains a list of width values, where the width value is a float value with units defined in the technology file. Each width value is for each wire in this wire pattern. The number of wires in this pattern is derived from the number of nets specified in

create_pg_composite_pattern or **set_pg_strategy** command. If the number of width values is smaller than the number of nets, the last width value is assumed for the extra nets. The width can also be specified by evaluating the variable *var* in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable. By default, the minimum width is used.

-pitch {*x_value y_value*} | *@var*

Specifies the pitch values for the wire pattern. {*x_value y_value*} specifies the horizontal and vertical pitches when applying to a wire pattern. If only one pitch value is specified, the value is applied to both horizontal and vertical pitches if applicable. The pitch value can also be specified by evaluating the *var* variable in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable. By default, the pitch is 0 for both horizontal and vertical pitches, and the wire pattern is not repeated.

-spacing minimum | interleaving | *value_list* | *@var*

Specifies the spacing between wires of the wire pattern. **minimum** refers to the minimum spacing between wires defined in the technology file based on the wire width. **interleaving** keyword means that distance between adjacent center lines of wires is equal. *value_list* contains a list of spacing values, where the spacing value is a float-type value with unit defined in the technology file. Each spacing value is for the adjacent wires in the pattern. For example, the first spacing value is for the first and the second wires. The number of wires in this pattern is derived from the number of nets specified in the **create_pg_composite_pattern** or **set_pg_strategy** command. If the number of spacing values is smaller than the number of nets minus one, the last spacing value is used for the remaining nets. The spacing can also be specified by evaluating the *var* variable in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable. By default, the minimum spacing is used if this option is not specified.

-trim true | false | *@var*

Specifies the trim option for this wire pattern. The "true" keyword means to remove one-end floating or dangling wires while "false" means to keep one-end floating or dangling wires. You can also specify the setting by evaluating the *var* variable in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable. By default, the value is "false".

-track_alignment track | half_track | *@var*

Specifies the track alignment option of this wire pattern. It can be specified by track or half_track. It can also be specified by evaluating the *var* variable in the **-parameters** option. The at character (@) character is used as a prefix to evaluate the variable.

-mask *mask_value*

Specifies the mask option for this wire pattern. You can specify either a keyword value or evaluate the *var* variable in the **-parameters** option. Valid keywords are **mask_one**, **mask_two**, **mask_three**, **mask_one_mask_two_alternate**, and **mask_two_mask_one_alternate**. To evaluate the *var* variable, use the at character (@) as a prefix to the variable name. The **mask_one_mask_two_alternate** keyword means that wires with an odd index have the mask_one color and wires with an even index have the mask_two color.

By default, the wire pattern is not colored.

-mask_constraint *constraint_name*

Specifies the mask constraint for the wire pattern. The PG mask constraint is defined by **set_pg_mask_constraint** command. The PG wires derived by this pattern will be checked against the specified constraint. It the constraint is not satisfied, the wires will not be created with error message issued. By default, PG mask constraint is not checked.

-parameters *var_list*

Specifies the list of parameters. Each parameter can be evaluated and used as the argument for other options in this command. This allows the pattern to be programmed and reused by passing different parameters in different situations. Each *var* parameter in the *var_list* list is a string. The at character (@) character is used as a prefix to evaluate the parameter, for example, *@var*. See the example section for more details.

DESCRIPTION

Creates a power ground wire pattern. This pattern can be used to create a composite pattern using the **create_pg_composite_pattern** command, or instantiated in a design by the **set_pg_strategy** command to create a power ground mesh.

A wire pattern contains a group of wires. All wires in this pattern share the same layer, direction, trim option, and low end and high end. The center of the first wire can be specified. Width and spacing can be specified for each wire and between adjacent wires, respectively. The number of wires can be derived from the number of nets specified in **create_pg_composite_pattern** or **set_pg_strategy** command. The wire pattern is not repeated while the pitch can be specified in **create_pg_composite_pattern** or **set_pg_strategy** command when using this wire pattern.

EXAMPLES

The following example creates a wire pattern M3_vertical_strap, where layer is M3, direction is vertical, low and high end extension is to routing area boundary. The width and spacing are specified by evaluating variables "width_list" and "spacing_list", respectively.

```
prompt> create_pg_wire_pattern \
   M3_vertical_strap \
   -parameters {width_list spacing_list} \
   -layer M3 \
   -direction vertical \
   -width {@width_list} \
   -spacing {@spacing_list} \
   -extend_low boundary \
   -extend_high boundary
```

The following example creates a wire pattern m5_segment, where the wire is a horizontal segment at layer M5 with width 1, and the length is specified by a variable "len".

```
prompt> create_pg_wire_pattern \
   m5_segment \
   -parameters {len} \
   -layer M5 \
   -direction horizontal \
   -low_end_reference_point 0 \
   -high_end_reference_point {@len} \
   -width 1
```

MORE EXAMPLES

For more example, please start GUI and invoke the following command. **gui_show_task -task "Design Planning:PG Planning->Examples->Overview"**

SEE ALSO

compile_pg(2)
create_pg_composite_pattern(2)
create_pg_macro_conn_pattern(2)
create_pg_ring_pattern(2)
create_pg_std_cell_conn_pattern(2)
remove_pg_patterns(2)
report_pg_patterns(2)
report_pg_strategies(2)
set_pg_strategy(2)
set_pg_mask_constraint(2)

create_pin

Creates one or more pins on a cell.

SYNTAX

collection **create_pin**
[-design *design*]
[-direction in | out | inout]
pin_names

Data Types

design collection
pin_names list

ARGUMENTS

-design *design*

Specifies the design in which to create the pin(s). If no design is specified, the pin(s) are created in the current design.

-direction in | out | inout

Specifies the direction of the pins. By default, the command creates pins with direction **in**.

pin_names

Specifies the names of the pins to create. Each pin name must be unique.

DESCRIPTION

This command creates pins of cells in the current design (unless otherwise specified by -design). It creates only scalar (single-bit) pins. One pin is created for each of the names listed. If a pin with the specified name already exists on the cell, the command displays an error message.

When the **create_pin** command creates pins, they are not connected. To establish a connection, you can use the **connect_net** command.

EXAMPLES

The following example creates pins named *data1* and *data2* on cell u14.

```
prompt> create_pin -direction in u14/data1
{"u14/data1"}
prompt> create_pin -direction out u14/data2
{"u14/data2"}
```

The following example creates input pins named *xg1* and *xg2* on cell u23/u8/u1.

```
prompt> create_pin {u23/u8/u1/xg1 u23/u8/u1/xg2}
{"u23/u8/u1/xg1", "u23/u8/u1/xg2"}
```

The following example creates pins named *usr1* and *usr2* on cell u5/u7.

```
prompt> current_instance u5/u7
u5/u7
prompt> create_pin -direction in {usr1 usr2}
{"u5/u7/usr1", "u5/u7/usr2"}
```

SEE ALSO

connect_net(2)
disconnect_net(2)
get_pins(2)
remove_pins(2)

create_pin_blockage

Creates a pin blockage in the current design.

SYNTAX

int **create_pin_blockage**
-boundary { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects
[-layers *layer_list*]
[-name *pin_blockage_name*]
[-cell *cell_name*]
[*object_list*]

Data Types

llx	float
lly	float
urx	float
ury	float
x	float
y	float
geometric_objects	collection
layer_list	list
pin_blockage_name	string
cell_name	string or collection
object_list	list

ARGUMENTS

-boundary { {{*llx lly*} {*urx ury*}} | {{*x y*} {*x y*} {*x y*} {*x y*}...} } | geometric_objects

Specifies the boundary coordinates of the pin blockage. The boundary can be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates as {*llx lly*} {*urx ury*}. A polygon is specified by its points as {*x y*} {*x y*} {*x y*} {*x y*} and so on.

Polygons may also be specified as the combined area of a mixed collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

When specifying the boundary polygon as a collection of physical objects, the resulting area must resolve to a single, connected polygon. It is an error to specify a collection of objects with a combined area that resolves to multiple polygons.

-layers *layer_list*

Specifies a list of layers for the pin blockage. Pins will not be assigned within the blockage area on these layers. If no layer is specified, the pin blockage applies to all layers and pins will not be assigned within the blockage area on all layers.

-name ***pin_blockage_name***

Specifies the name of the pin blockage.

-cell ***cell_name***

Specifies the physical cell where the pin blockage is to be added. The pin blockage is created in the reference block for the cell using the coordinate system of the cell argument's top block. The cell must reference a block, not a library cell, unless this command is executed in the library manager. In the library manager, pin blockages can be created in library cells. When not specified, the pin blockage is created in the current block.

-feedthrough_only

Specifies that this pin blockage prevents feedthrough pins from being placed in the specified area.

object_list

Specifies a list of physical pins, nets, or ports to associate with this pin blockage. The object list must be homogeneous; it cannot contain a mixture of different object types. Only the specified pins, nets' pins, and ports will avoid this pin blockage. If this option is not specified, all pins and ports will avoid this pin blockage. A pin, net, or port can be associated with multiple pin blockages. Pin assignment will avoid all pin blockages associated with a particular pin, net, or port.

DESCRIPTION

The **create_pin_blockage** command creates a pin blockage with the specified boundary in the current design. The pin blockage can be associated with a list of physical pins, nets, or ports. The list must be homogeneous; it cannot contain a mixture of object types. Each pin, net, and port can be associated to multiple pin blockages.

All associated pins, pins of associated nets, and associated ports must be placed outside of the blockage. If no objects are associated, all pins and ports will be placed outside of the blockage during pin assignment.

EXAMPLES

The following example creates a pin blockage named 'PB_1' with rectangular boundary that applies to all layers.

```
prompt> create_pin_blockage -boundary {{100 100} {200 200}} \
  -name PB_1
```

The following example creates a pin blockage with a rectilinear boundary that applies to the M1 and M2 layers only.

```
prompt> create_pin_blockage -boundary {{0 0} {2000 0} \
  {2000 2000} {1000 2000} {1000 1000} {0 1000}} -layers {M1 M2}
```

The following example creates a pin blockage with a rectangular boundary that is associated with physical pins.

```
prompt> set pins [get_pins -physical_context cell1/pin*]
prompt> create_pin_blockage -boundary {{100 100} {200 200}} $pins
```

SEE ALSO

add_to_pin_blockage(2)
get_pin_blockages(2)
remove_from_pin_blockage(2)
remove_pin_blockages(2)
report_pin_blockages(2)

create_pin_bus

Creates one pin_bus.

SYNTAX

collection **create_pin_bus**
[-design *design*]
[-create_pins]
[-block *block*]
-direction in | out | inout
-cell_bus *cell_bus*
pin_bus_names

Data Types

design collection
pin_bus_name string

ARGUMENTS

-design *design*

Specifies the top-level design for finding objects. If this is not specified, objects are found in the current design.

-create_pins

Creates pin bus members if they do not exist.

-block *block*

Specifies the block where the pin bus is to be added. If specified, **create_pin_bus** has the same effect as **edit_module** in which the module is changed and changes are applied to all module occurrences.

-direction in | out | inout

Specifies the direction of the pin_bus members.

-cell_bus *cell_bus*

Specifies the cell_bus where the pin bus has to be created.

pin_bus_name

Specifies the pin bus name which has to be created. Doesn't support patterns containing hierarchial separators.

DESCRIPTION

This command creates a pin bus, collects or creates pin bus members, and puts them into the pin bus created.

Width of created pin bus will be same as cell bus on which it is created.

The command returns the created pin_bus(as a collection), an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

EXAMPLES

The following example creates pin bus named bus1 from pins 1 to 5 as bus[1]/bus1, bus[2]/bus1, bus[3]/bus1 and so on. The width of the pin bus is the same as the cell bus on which it is created.

```
prompt> create_pin_bus -cell_bus bus bus1
{"bus/bus1"}
```

The following example first creates pins, then creates a pin bus named bus/bus2 from the created pin bus[1]/bus2, bus[2]/bus2, bus[3]/bus2 and so on.

```
prompt> create_pin_bus -cell_bus bus bus2 -create_pins
{"bus/bus2"}
```

SEE ALSO

get_pin_buses(2)
remove_pin_buses(2)
report_pin_buses(2)

create_pin_constraint

Creates individual and bundle pin constraints.

SYNTAX

status **create_pin_constraint**
-type individual | bundle
[-cells *cells*]
[-layers *layers*]
[-pin_spacing_tracks *spacing*]
[-pin_spacing_distance *distance*]
[-allow_feedthroughs true | false]
[-sides *side_numbers*]
[-width *width*]
[-length *length*]

[-nets *nets*]
[-pins *pins*]
[-ports *ports*]
[-exclude_sides *exclude_side_numbers*]
[-offset *offset_distance*]
[-off_edge]
[-location {*x y*}]

[-bundles *bundles*]
[-keep_pins_together true | false]
[-range {*start end*}]
[-bundle_order ordered | increasing | decreasing | equal-distance]
[-self]

Data Types

cells collection cells
layers collection of layers
spacing integer
distance distance
side_numbers collection of integers
width string
length string
nets collection of nets
pins collection of pins
ports collection of ports
offset_distance string
bundles collection of net bundles
range string
x float
y float

ARGUMENTS

Note the first set of arguments applies to both individual and bundle pin constraints. The second set of arguments applies to only individual pin constraints, and the third set of arguments applies only to bundle pin constraints. If the user provides a type-specific argument for the incorrect type the command will return an error.

-type individual | bundle

Specifies the type of pin constraint to create.

-cells *cell_collection*

Limits the constraints to the specified cells. The cells must correspond to sub-levels of physical hierarchy in the top design. When creating individual pin constraints, this argument is only valid when used with **-nets**.

-layers *metal_layers*

Specifies the set of metal layers allowed for pin placement. The argument is a list of metal layer names as returned by the **get_layers** command.

By default, the metal layers from M2 to the maximum metal layer (specified earlier during floorplan flow) are allowed for pin placement. If the maximum metal layer has not been specified, the maximum available metal layer specified in the technology file is used as the default. If the maximum layer specified is beyond the maximum metal layer for the current design, then the maximum layer in the technology file is used.

-pin_spacing_tracks *spacing_number*

Specifies the minimum number of wire tracks between adjacent pins or between a pin and a preroute wire that cuts across a block edge in the normal direction. Pins that are created are always snapped to wire tracks. The default pin-to-pin spacing is one wire track. The spacing number must be 0 or a positive integer.

This option is mutually exclusive with **-pin_spacing_distance**.

-pin_spacing_distance *spacing_distance*

Specifies the minimum distance between adjacent pins or between a pin and a preroute wire that cuts across a block edge in the normal direction. Pins that are created are always snapped to wire tracks.

This option is mutually exclusive with **-pin_spacing_tracks**.

-allow_feedthroughs true | false

Specifies whether feedthrough ports can be created in pin placement flow. When true, the global router can create feedthrough routing on a block cells for the specified nets. When a feedthrough is created on a top-level net, the net is split into a set of new top-level nets and child-level nets if feedthrough nets are created for the child nets. Directions are assigned for the new feedthrough ports, and the netlist is modified to accommodate the new ports in the block cell.

When used for individual pin constraints, this option can only be specified together with the **-nets** option and cannot be used together with **-cells** option. To set feedthrough constraints on individual block cells for selected nets, create a pin constraints file and use the **read_pin_constraints** command.

This option can also be used to set feedthrough allowance for the push-down into a block of pre-routed nets using **push_down_objects**. That command will check this pin constraint, and if found to be set, and set to true, will create feedthroughs based upon the pre-route multiple crossing of a block's boundary.

For individual pin constraints, this option is only applicable with the **-net** argument. Also, for individual pin constraints, this option is mutually exclusive with the **-cells** argument.

-sides *side_numbers*

Specifies one or more block sides on which the pin must be placed.

A side number is a positive integer that starts at one, and increments by one as you proceed clockwise around each edge in the shape. Given any rectilinear shape, the left-most vertical edge is the starting edge (side number 1). If there are multiple left-most edges, then the edge 1 is the lowest left-most edge. You cannot specify a value of 0 for this option.

For individual pin constraints, this option is mutually exclusive with **-exclude_sides** option.

-width *pin_width*

Specifies the width of the pin. The width is perpendicular to the layer's preferred routing direction. The syntax can be a single real number to indicate the width is applied to all allowed layers (referred to as common pin width). Or alternatively, the width can be specified as a list of layer-real pairs as the following (referred to as per layer pin width):

-width {{M2 0.2} {M3 0.2} {M4 0.3} {M5 0.3}}

It means that if the pin is to be placed on layer M2 or M3, its width should be 0.2 and if on M4 or M5, 0.3. On other layers, use the default width defined by the tech file.

If you first set the pin width to be common pin width, then set the pin width to be per layer width with a second **create_pin_constraint** command, then the width specified in the second command will override the first. Likewise, if you first set the pin width to be per-layer width, then set the pin width to be a common pin width with a second **create_pin_constraint** command, then the width specified in the second command will override the first.

However, if you first set the pin width to be per layer, then in a second command sets the pin width again to be per layer but with different layers or different values, then the combined per layer pin width will be the final constraints. For example, consider the following example:

```
create_pin_constraint -bundles {bundle1} -width 0.3
create_pin_constraint -bundles {bundle1} -width {{M2 0.3} {M3 0.3}}
```

The first common pin width constraint will be overridden by the second per layer pin width. The pin width should be 0.3 on layer M2 or M3, but default width on all other layers.

And for the following example:

```
create_pin_constraint -bundles {bundle1} -width {{M3 0.4} {M4 0.4}}
create_pin_constraint -bundles {bundle1} -width {{M2 0.3} {M3 0.3}}
```

The pin width should be 0.3 on layer M2 or M3, 0.4 on layer M4, and default width on all other layers.

-length *pin_length*

Specifies the length of the pin. The length is in parallel to the layer's preferred routing direction. The syntax can be a single real number to indicate the length is applied to all allowed layers (referred to as common pin length). Or alternatively, the length can be specified as a list of layer-real pairs as the following (referred to as per layer pin length):

-length {{M2 0.2} {M3 0.2} {M4 0.3} {M5 0.3}}

It means that if the pin is to be placed on layer M2 or M3, its length should be 0.2 and if on M4 or M5, 0.3. On unspecified layers, the pin length is derived from the pin width. Refer to the manpage of **place_pins** for details.

-pins *pins*

Specifies the pins that receive the constraints. Specify the full pin name from top-level design.

-nets *nets*

Applies the constraints only to the nets specified in *nets*.

-ports *ports*

Specifies the ports that receive the constraints.

-exclude_sides *exclude_side_numbers*

Specifies the block edges on which pins cannot be placed. The *exclude_side_numbers* argument is a list of positive integers.

A side number is a positive integer that starts at one, and increments by one as you proceed clockwise around each edge in the shape. Given any rectilinear shape, the left-most vertical edge is the starting edge (side number 1). If there are multiple left-most edges, then the edge 1 is the lowest left-most edge. You cannot specify a value of 0 for this option.

When creating individual pin constraints, this option is mutually exclusive with **-sides** option.

-offset *offset_distance*

Specifies the distance in microns between the starting or ending point of a specified edge and the pin's center location. The starting point is the location where the edge begins as you proceed clockwise around the shape. The ending point is the location where the edge end as you proceed clockwise around the shape. The starting point of edge 1 is the lowest point of the edge, the ending point of edge 1 is the highest point of the edge the starting point of edge 2 is the leftmost point of the edge, and so on. Positive value means offset distance calculates from starting point of a specified edge and the pin's center location, vice versa for negative value.

A range can also be specified for this option. Specify range start and range end values within which the pins are allowed to place. The start and end values refer to the distances as measured to the starting point of the edge in microns (or the default unit). Negative value for range refer to the distances as measured to the ending point of the edge in microns (or the default unit). For example,

-offset {10 40}

You must specify the side information together with the offset. However, offset cannot be specified with multiple sides.

-off_edge

Specifies whether the pin is created on the block boundary. This option is mutually exclusive with the **-sides** and **-offset** options.

-location *{x y}*

Specifies the x- and y-location in the top-level design where the pin should be created. This option is mutually exclusive with the **-sides** and **-offset** options. When used together with the **-off_edge** option, the pin is placed on the closest wiretrack to the specified location. Note that in this case there will be no legalization of pin placement for off edge pins, as it is meant to be a basic placement where user is expected to give the legal location. If the location constraint is a hard constraint set by the **set_block_pin_constraints** command, the center of the pin is placed at the exact location specified by this option. If you do not specify the **-off_edge** option, the pins are placed on the edge that is closest to this location.

-bundles *bundles*

Limits the constraints to the specified net bundles. The bundles must be in the top-level of the current design. If not specified, the command applies the constraints to all net bundles.

-keep_pins_together true | false

Specify whether bundle pins are kept together. When this option is true, all bundle pins on the same block are placed on the same layer (hard constraint) and no other pins should be placed between the bundle pins (soft constraint). In certain cases, the command might violate the soft constraint and insert other pins between the bundle pins. The default value is false.

-range {*start end*}

Specifies the range of positions on a side within which the bundle pins must be placed. The positive *start* and *end* values refer to the distances as measured to the starting point of the edge in microns (or the default unit). The negative *start* and *end* values refer to the distances as measured to the ending point of the edge in microns (or the default unit). Postive *start* is the position that is

closest to the edge's starting point, and postive *end* is the position that is farthest to the edge's starting point. Vice versa for negative *start* and *end* positions. The starting point of an edge is the vertex of the edge where the edge begins as you proceed clockwise around the block's boundary from the lowest vertex of the leftmost edge. If there are more than one leftmost edges, then from the lowest one.

The following simple drawing shows an example of how postive *start* and *end* are measured on a rectangular block on each of the sides.

```
 +---start-------end------+
 |                |
 |                start
end                 |
 |                |
 |                end
start               |
 |                |
 +------end-----start-----+
```

The following simple drawing shows an example of how negative *start* and *end* are measured on a rectangular block on each of the sides.

```
 +---end-------start------+
 |                |
 |                end
start               |
 |                |
 |                start
end                 |
 |                |
 +------start-----end-----+
```

-bundle_order ordered | increasing | decreasing | equal-distance

Specifies the relative placement order of the pins connected to the bundle nets. Note that the order refers to the net order in the bundle, not the pins on the block. The bundle order is measured as seen at the current top design. Pin placement considers this option only when option **-keep_pins_together** is set to true. If two or more bundles connecting the same collection of MIB pins but with conflict orders, only one of them can be honored.

- ordered: Pins are ordered according to the net order in the bundle. For vertical block sides, bundle pins can be placed either from bottom to top or from top to bottom. For horizontal sides, bundle pins can be placed either from left to right or from right to left. The orders on each edges are chosen such that for the bundle nets connect the bundle pins from one edge to another edge, there is no net crossing, i.e. river turn style.

  In the following example, bundle pin constraint is specified as *ordered* so that the nets connecting two bundle pins are of the river turn style, i.e. no crossing.

  ```
  prompt> create_pin_constraint -type bundle -layers {M3} -bundles bundle1 \
    -bundle_order ordered -keep_pins_together true
  ```

- increasing: Pins are ordered according to the increasing net order in the bundle. For vertical block sides, bundle pins are placed from bottom to top. For horizontal block sides, bundle pins are placed from left to right.

  In the following example, bundle pin constraint is specified on a rectangular block so that bundle pins are placed on side 2 (horizontal edge) of block. Pins are placed from left to right (Least significant bit is the leftmost; Most significant bit is the right most on side2).

  ```
  prompt> create_pin_constraint -type bundle -bundles bundle1 -sides 2 \
    -bundle_order increasing -keep_pins_together true
  ```

- decreasing: Pins are ordered according to the decreasing net order in the bundle. For vertical block sides, bundle pins are placed from top to bottom. For horizontal block sides, bundle pins are placed from right to left.

  In the following example, bundle pin constraint is specified on a rectangular block so that bundle pins are placed on side 2 (horizontal edge) of block. Pins are placed from right to left (Most significant bit is the leftmost; Least significant bit is the rightmost).

  ```
  prompt> create_pin_constraint -type bundle -bundles bundle1 -sides 2 \
    -bundle_order decreasing -keep_pins_together true
  ```

- equal-distance: Pins are ordered as with **-bundle_order ordered**, but the pin ordering for two edges is chosen to create the same Manhattan distance between each corresponding pair of pins.

  In the following example, bundle pin constraint is specified as *equal-distance* so that the Manhattan distacnes of each bits of the bundle are the same for bundle *bundle1*.

  ```
  prompt> create_pin_constraint -type bundle -layers {M3} -bundles bundle1 \
    -bundle_order equal-distance -keep_pins_together true
  ```

  Note that when setting bundle pin constraints on pins of multiple instantiated blocks (MIB), as the orientation of MIBs can be different, the tool chooses one of the MIBs randomly to honor the constraints. Therefore the bundle constraints could be not honored for the other MIBs.

  When option **-keep_pins_together** is set to true but option **-bundle_order** is not specified, the tool will choose from either **increasing** or **decreasing** based on alignment, abutment and wire length cost. The bundle order might also be decided by already placed or fixed bundle pins on this bundle due to alignment or abutment.

-self

Applies the constraints to the top-level block. You can combine the **-cells** and **-self** options. If both the **-cells** and **-self** options are unspecified, the constraints apply to all block cells.

DESCRIPTION

This command constrains the placement of net bundles, individual pins, individual ports, or individual nets on a physical design block. The pin placement constraints are saved in the design database. The constraints are honored by the **place_pins** command.

The **create_pin_constraint** command is additive. The constraint values set in previous calls are retained, unless they are explicitly overridden by subsequent calls.

The **set_editability** command can enable or disable the **create_pin_constraint** command for specified blocks or levels of physical hierarchy. This command has no effect on the blocks that are disabled with the **set_editability** command.

- Global constraints: Constraints apply to all bundles and all block instances.
- Specific bundle constraints: Constraints apply to a specific bundle and all block instances.
- Specific block constraints: Constraints apply to all bundles and a specific block instance. Note that this can apply to the top-level block when you specify the *-self* option.
- Bundle and block pair constraints: Constraints apply to a specific bundle and a specific block instance. Note that this can apply to the top-level block when you specify the *-self* option.

Upon success, the command returns a collection of pin constraints containing the newly created pin constraint.

EXAMPLES

The following example sets the allowed pin layers to METAL2 and METAL3 for bundle BUNDLE_0. This creates a specific bundle constraint.

```
prompt> create_pin_constraint -type bundle -layers [get_layers METAL2 \
  METAL3] -bundles [get_bundles BUNDLE_0]
```

The following example enforces ordering on all bundle pins of cell BLOCK1. This creates a specific block constraint.

```
prompt> create_pin_constraint -type bundle -bundle_order ordered \
  -cells [get_cells BLOCK0] -keep_pins_together true
```

The following example enforces ordering and minimum pin spacing for bundle BUNDLE_0. This creates a specific bundle constraint.

```
prompt> create_pin_constraint -type bundle -bundle_order ordered \
  -pin_spacing_tracks 2 -bundles [get_bundles BUNDLE_0] -keep_pins_together true
```

The following example sets the allowed pin layers for net clk to M2 and M3.

```
prompt> create_pin_constraint -type individual -layers [get_layers {M2 M3}] \
  -nets [get_nets clk]
```

SEE ALSO

place_pins(2)
push_down_objects(2)
report_pin_constraints(2)
remove_pin_constraints(2)
get_pin_constraints(2)
set_block_pin_constraints(2)
read_pin_constraints(2)
set_editability(2)

create_pin_guide

Creates a pin guide in the current design.

SYNTAX

int **create_pin_guide**
-boundary { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects
[-layers *layer_list*]
[-parents *cell_list*]
[-name *pin_guide_name*]
[-exclusive]
[-pin_spacing *number_of_wire_tracks*]
[-cell *cell*]
object_list

Data Types

llx	float
lly	float
urx	float
ury	float
x	float
y	float
geometric_objects	collection
layer_list	list
cell_list	list
pin_guide_name	string
number_of_wire_tracks	int
cell	string or collection
object_list	list

ARGUMENTS

-boundary { {{*llx lly*} {*urx ury*}} | {{*x y*} {*x y*} {*x y*} {*x y*}...} } | geometric_objects

Specifies the boundary coordinates of the pin guide. The boundary can be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates, for example, {*llx lly*} {*urx ury*}. A polygon is specified by its points, for example, {*x y*} {*x y*} {*x y*} {*x y*} and so on.

Polygons can also be specified as the combined area of a mixed collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every

shape in the layer.

When specifying the boundary polygon as a collection of physical objects, the resulting area must resolve to a single, connected polygon. It is an error to specify a collection of objects with a combined area that resolves to multiple polygons.

-layers *layer_list*

Specifies a list of layers for the pin guide. Pins will be placed only on these layers. If layers are not specified, pins can be placed on any layer.

-parents *cell_list*

Specifies a list of block cells to which this pin guide applies. If not specified, this pin guide is applied to the current design.

-name *pin_guide_name*

Specifies the name of the pin guide.

-exclusive

Specifies that this pin guide is exclusive. This prevents the placement of terminals within the pin guide, except for pins, pins of nets, and ports associated to this pin guide.

-pin_spacing *number_of_wire_tracks*

Specifies the number of wire tracks between pins within this pin guide. When this option is specified, the pin placement command will place pins, within the pin guide, such that the pin minimum spacing is equivalent to the number of specified wire tracks.

-cell *cell*

Specifies the physical cell where the pin guide is to be added. The pin guide is created in the cell's reference block using the coordinate system of the cell argument's top block. The cell must reference a block, not a library cell, unless this command is executed in the library manager. In the library manager, pin guides can be created in library cells. When not specified, the pin guide is created in the current block.

object_list

Specifies a list of physical pins nets, or bundles of nets to associate with this pin guide if the parents for the pin guide are block cells. Specifies a list of physical ports, nets, or bundles of nets to associate with this pin guide if the parent for the pin guide is the current design. The object list must be homogeneous; it cannot contain a mixture of different object types. During pin assignment, the terminals generated for these objects are constrained to the pin guide bounding box. Currently bundles of supernets is not supported. If bundle of supernets is specified then it will be skipped with a warning message.

DESCRIPTION

The **create_pin_guide** command enables you to create a pin guide with the specified boundary in the current design. The pin guide has one or more parent cells, to which the pin guide is applied. The parents of the pin guide can be the current design or a list of block cells. During pin assignment, the terminals generated for these objects are constrained to the pin guide bounding box.

If the parent of the pin guide is the current design, the pin guide can be associated with a list of physical ports or nets. If the parents of the pin guides are block cells, the pin guide can be associated with a list of physical pins or nets. The list must be homogeneous; it cannot contain a mixture of object types. Each pin, net, and port can be associated to multiple pin guides.

EXAMPLES

The following example creates a pin guide named 'PB_1' with rectangular boundary that applies to all layers of the current design. It is associated with physical ports of the current design.

```
prompt> create_pin_guide -boundary {{100 100} {200 200}} \
  -name PB_1 [get_ports clk*]
```

The following example creates a pin guide with a rectilinear boundary that applies to the M1 and M2 layers only in the current design. It is associated with physical nets in the current design.

```
prompt> create_pin_guide -boundary {{0 0} {2000 0} {2000 2000} \
  {1000 2000} {1000 1000} {0 1000}} -layers {M1 M2} [get_nets n*]
```

The following example creates a pin guide with a rectangular boundary that is associated with physical pins of the macro cell 'macroA'.

```
prompt> set macro macroA
prompt> create_pin_guide -boundary {{100 100} {200 200}} \
  -parents [get_cells $macro] [get_pins $macro/*]
```

SEE ALSO

add_to_pin_guide(2)
get_pin_guides(2)
remove_from_pin_guide(2)
remove_pin_guides(2)
report_pin_guides(2)

create_placement

Performs coarse placement on the current design.

SYNTAX

status **create_placement**
[-floorplan]
[-host_options *host_option_name*]
[-use_seed_locs]
[-effort very_low | low | medium | high]
[-timing_driven]
[-buffering_aware_timing_driven]
[-congestion]
[-congestion_effort low | medium | high]
[-incremental]

Data Types

host_option_name string

ARGUMENTS

-floorplan

Specifies that floorplanning placement should be used. This includes macro placement and special effort levels which are tuned for speed. There are a number of special app option controls for floorplanning. To get a list of them use **report_app_options plan.place.*** and **report_app_options plan.macro.***. One app option which you must be aware of is **plan.place.auto_generate_blockages**. This controls whether soft blockages are automatically created in thin channels between hard macros and block boundary. This is useful for congestion reduction. See man page of this app option for more details.

-host_options *host_option_name*

Specifies that the given host option should be used for distributed processing. Host options are set with the **set_host_options** command. Note that not all operations performed by the **create_placement** command support distributed processing. The effect of this option depends on the design and the other option settings.

-use_seed_locs

Specifies that the current placement should be used as a seed for placement.

-effort very_low | low | medium | high

Specifies the CPU effort level for coarse placement. The default effort level is medium for floorplanning placement, and high for all other cases.

-timing_driven

Enables timing-driven placement. When used with the -floorplan option, you can adjust the behavior of the command with the plan.place.timing_driven_mode application option.

-buffering_aware_timing_driven

Enables timing-driven placement using an approximate timing model that estimates the effects of buffering long nets and high fanout nets later in the flow. The netlist is not changed. Intended for use as an initial placement step that gives a better starting point for later timing optimizations.

-direct_timing

Obsolete. Use the **-timing_driven** option instead.

-congestion

Enables congestion-driven placement mode.
When used with the **-floorplan** option, you can adjust the behavior of the command with the **plan.place.congestion_driven_mode** application option.

-congestion_effort low | **medium** | **high**

Specifies the effort level for congestion mode. The default effort level is medium. Expect a significant increase in runtime for high effort. This option can only be used with the **-congestion** option.

-incremental

Performs incremental placement on the design. Incremental placement tries to keep cells close to their current location. The **-incremental** option automatically enables the **-use_seed_locs** option. By default, the command can move cells far from their current location to improve placement QoR.
When **-incremental** is used in combination with **-floorplan**, the macro placement will be legalized while trying minimizing the macro movements. Also, the standard cell placement will be updated in incremental fashion. When also combined with the **-congestion** option, the command expects that the current placement is legal and that an up-to-date congestion map exists. This congestion map is used to increase the channel sizes between the macros in congested areas. Again, the standard cell placement is updated in an incremental fashion. See also the man page for the **plan.place.congestion_driven_mode** application option.

DESCRIPTION

The **create_placement** command performs coarse placement on the current design. With the **-floorplan** option, this includes the placement of hard macros as well as standard cells. To avoid placing hard macros, mark the hard macros as fixed with the **set_attribute** command.

EXAMPLES

The following example runs the **create_placement** command in timing-driven mode:

```
prompt> create_placement -effort high -timing_driven
```

SEE ALSO

check_host_options(2)
legalize_placement(2)
read_def(2)
report_app_options(2)
set_attribute(2)
set_host_options(2)
write_def(2)
plan.place.congestion_driven_mode(3)
plan.place.auto_generate_blockages(3)

create_placement_attraction

Advise coarse placer on relative placement of modules

SYNTAX

collection **create_placement_attraction**
[-name *attraction_name*]
[-region coord_list]
[-effort very_low | low | medium | high]
[-exclude_buffers none | edge | all]
[-add_internal_logic false | true]
[*cell_list*]

Data Types

attraction_name string
coord_list list
cell_list list

ARGUMENTS

-name *attraction_name*

Specifies the name of the placement attraction.

-region coord_list

Specifies the region for the placement attraction. The region may be exactly one of the following: unset (floating), point, line, rectangle. The argument takes a coordinate list and parses it to determine the desired region. If the user does not specify a value the region is unset. A coordinate list with one point corresponds to a point region. A coordinate list with two co-linear points that form an axis-parallel line corresponds to an edge region. A coordinate list with two non-co-linear points corresponds to an area region with coordinates specifying the lower left and upper right point of the rectangle. For more details, refer to the description section of this man page.

-effort very_low | low | medium | high

The **-effort** option allows you adjust how strong your attraction suggestion will be. Please see the section on "Setting the effort" below.

-exclude_buffers none | edge | all

The placer will apply a filter to the buffers and inverters in the attraction constraint. If all you specify "all", the placer will exclude all of the specified buffers and inverters from the effect of the attraction. If you specify "edge", the placer will exclude all buffers and inverters that are on timing paths that lead to logic that is outside of the attraction group. This is useful so that you don't overconstrain buffer chains which should not have attraction to a central location. If you specify "none", then the buffers and

inverters will not be excluded from the effect of the attraction. The default setting is "edge".

-add_internal_logic false | true

When specified, the placer will try to work on extra cells, in addition to the cells specified in your attraction. In particular, the placer will include logic cells that occur on timing paths between two flops that are already declared in the attraction. This is useful when you have ungrouped your logic hierarchies and run initial optimization. In general, the instance names of ungrouped logic are not maintained, which makes it difficult to put that logic into an attraction. But since flop names are generally maintained, you can put selected flops in the attraction and use this command option to fill in the intervening random logic. The default value is true.

cell_list

Specifies a list of cells to be included in the attraction constraint. Please see the section on "Specifying your cell list" below. Some or all of the cells in the list may correspond to logic hierarchies in your block. If a logic hierarchy is specified, the all of the leaf cells under that hierarchy are also implicitly included in the attraction constraint.

DESCRIPTION

When placing a block, there is often significant latitude for configuring individual of logic modules within that placement. For example, a particular logic module may be placed on the left or on the right -- with little difference in the overall placement quality. The **create_placement_attraction** command allows you to suggest how modules (or any arbitrary group of cells) should be placed -- both relative to each other and relative to the floorplan.

The **create_placement_attraction** command is a much "softer" placement constraint than the move bound or group bound constraints created by the **create_bound** command. The placement attraction is used to tell the placer about the module relationships that are important, without the need to strongly constrain every cell in each module. In general, the placement attractions will steer the placer in the right direction, but it will also trade off with other considerations like wire length and timing.

The placement_attraction constraint is meant act on modules or big groups of cells. If you are interested in constraining small groups of cells, you should be using the **create_bound** command. The placement_attraction can direct general placement, but it is generally too weak to force the placement of individual cells.

The effect of the **create_placement_attraction** command will be most pronounced in the *first placement* and will have a strong effect on where in a logic modules are placed. This makes placement attraction an excellent tool for "high level" communication with the placer. If you have run the placer and don't like where a module is located, this is how you tell the placer what you would like to see instead.

Placement attractions will not have a strong effect in subsequent (incremental or seed) placements. This is by design, so that the attractions will not block optimization of individual timing paths or congested areas. So if you are want to add new attractions to your block, you should restart your flow from *initial placement*. Note that some flows use a first placement from logic synthesis. If you use this flow, then the attractions should be set before the logic synthesis placement.

The **-region** option allows you to direct a module or group of cells to be near a given point in your floorplan. You can get different behaviors, depending on how you specify the coordinates to this option. If you specify a single point coordinate, cells will be attracted to that point. Unlike a move bound, there is no need to provide a "boundary". The placer will decide how close to the point individual cells should be. Cells with critical timing may drift very far away to improve a difficult path -- if that is appropriate.

If you specify a horizontal or veritical line to **-region**, modules or group of cells will attract near that line in your placement. It is presumed, that the given line will be near the edge of block or a macro, but that is not a requirement. Unlike a move bound, there is no need to provide a "boundary". The placer will decide how close to the line individual cells should be. Cells with critical timing may drift very far away to improve a difficult path -- if that is appropriate.

If you specify a box area to **-region** option, modules or group of cells will attract near that given area in your floorplan. The placer will decide how close to the area individual cells should be. Cells with critical timing may drift very far away to improve a difficult path -- if that is appropriate. Unlike a move bounds, there is no strong force to keep cells inside of the given box, only close. If you want to

force your cells to be mostly inside of box, use the **create_bound** command to create a move bound.

If you do not specify **-region**, then the specified cells will be kept together, but will not be constrained to a particular location in the floorplan. Unlike a group bound from the **create_bound** command, there will not be an overly strong grouping of the objects.

See the section on "Common usage" below for more details.

Setting the effort

The **-effort** option allows you adjust how strong your attraction suggestion will be. Stronger attractions will be given higher weight by the placer, but may result in some amount of over-constraining. For example, if you want to group two modules together, a high effort may hold the modules together too tightly. In general, you will want to use the lowest setting that delivers the desired result. The default setting is "low".

If you find that you need to use high effort to get the desired effect, you might want to think harder about why this is happening. If you need high effort, there is probably some competing constraint that is fighting what you want to do. For example, maybe the floorplan does not have enough space to accommodate what you want, where you want it. Or maybe there are many connections pulling your module in the opposite direction. If even high effort attraction does not work, you probably need to use a move bound to strongly force what you want.

Note that the effort levels of **create_placement_attraction** are not equivalent to those in **create_bound**. Bounds are much stronger.

Specifying your cell list

When creating a placement attraction, you need to specify the standard cells that participate in the attraction constraint. Most often, the cells will be found using the **get_flat_cells** command, with a regular expression that matches cells in a particular module. For example:

```
create_placement_attraction -name abc [get_flat_cells u_abc/*]
```

If you have preserved your logical hierarchy, the get_flat_cells command will give you all of the cells in your module. Alternatively, you can choose to simply put the logic hierarchy cell instance directly in your cell list. All contained leaf cells will be implicitly included along with the parent hierarchy.

If you have flattened some or all of your logical hierarchy, you might only find the sequential elements of a module. It is OK if you do not find all of the cells of your module. Often, setting attraction on only the sequential elements of a module will give the desired effect. The **-add_internal_logic** option (which is true by default) will help find the missing non-sequential cells.

It is often useful to use temporary collections of cells for each module that you want to constrain. The you can refer to the collections in your **create_placement_attraction** command. For example, you can create collections for module abc and module xyz:

```
set abc_cells [get_flat_cells u_abc/*]
set xyz_cells [get_flat_cells u_xyz/*]
```

After you do this you can make attractions between the modules like this:

```
create_placement_attraction -name abc_xyz "$abc_cells $xyz_cells"
```

Note that the same cells may participate in multiple placement attraction constraints. And fixed cells will be ignored.

Common usage

For the most part the placer is very good at placing modules within a block floorplan. The placer considers wire length, timing, and congestion when arranging modules within the floorplan. Placement attraction is best used only to adjust the default placement, not to dictate it. When you are first starting to work on a block, it is suggested that you don't use any placement attraction directives. This will give the placer the maximum flexibility to optimize your design.

As you find placement decisions you want to adjust, you can add attraction directives. But you should only add directives where they are needed to achieve the desired result. You may find that one added attraction on one module may have a "ripple effect" that improves other module placements. So avoid adding too many attractions, unless you absolutely need them. Using fewer attraction constraints will give the placer the maximum flexibility to optimize your design. Also, keep in mind that attractions only work well during the initial placement, not incremental placement. If you add a new attraction, start your placement again from scratch.

There are several ways to use placement attraction. We will describe a few of them below. Please note that there is a theme for the following examples. The placement_attraction constraint is meant act on modules or big groups of cells. If you are interested in constraining small groups of cells, you should be using the **create_bound** command. The placement_attraction can direct general placement, but it is generally too weak to force the placement of individual cells.

You can use **create_placement_attraction** to force the placement of a module in the floorplan. Usually, you would either use **-region** with a single point to direct the module placement to be in a corner or small area, or you would use **-region** with a horizontal or vertical line to direct a module to hug the side of the block or macro. Unlike using a move bound for this purpose, you do not have to mark out the exact boundary of the module -- just the general area.

You can use **create_placement_attraction** to influence the placement of a module near a set of RAMs. Usually, you would use **-region** to mark the area over the RAMs to pull the module close by. This is different than using a move bound from the **create_bound** command, which would tend to force the module on top of and in between the RAMs. For example, if you have a RAM array that occupies the area {{1000 1000} {1500 1500}}, then the following three constraints operate in very different ways:

```
# Place cells in between the RAMs:
create_bound -bound {{1000 1000} {1500 1500}}

# Attract the cells near a point at the center of the RAM array:
create_placement_attraction -region {{1250 1250}}

# Attract cells to be next to the RAM array:
create_placement_attraction -region {{1000 1000} {1500 1500}}
```

You can use **create_placement_attraction** without **-region** a single module to hold it together. Sometimes, when a module is not very "dense", other modules may overlap or come in the middle of your module. The attraction will pull the module together more tightly. Unlike using a group bound for this purpose, the attraction will not unduly squish the module into a particular shape.

You can use **create_placement_attraction** without **-region** to keep two or more modules in close proximity to each other. This is done by putting the cells of both modules into the same attraction group. You can use attractions with varying efforts to make more complicated groupings. For example, you can group an adder and multiplier closely together by using a "medium" effort attraction group. And then group both the adder and multiplier with other ALU units in a "low" effort attraction group.

You can use **create_placement_attraction** to influence the placement optimization of critical timing paths. For example, it is sometimes the case that timing can only be met when entire modules are carefully placed -- near a port, near a macro, or near each other. Using the **-region** options will guide placement near ports or macros. You can use **create_placement_attraction** without **-region** to keep modules together.

EXAMPLES

The following sets the attraction of cells from the "abc" module near a point. Note that two sets of curly brackets are required to make a valid coordinate list.

```
prompt> set abc_cells [get_flat_cells u_abc/*]
prompt> create_placement_attraction -name abc_near_ll \
 -region {{0 0}} $abc_cells
```

The following sets the attraction of cells from the "abc" module near a vertical edge.

```
prompt> set abc_cells [get_flat_cells u_abc/*]
prompt> create_placement_attraction -name abc_on_the_left \
 -region {{0 0} {0 1500}} $abc_cells
```

The following sets the attraction of cells from the "abc" module near the area occupied by L2 RAMs.

```
prompt> set abc_cells [get_flat_cells u_abc/*]
prompt> set l2_rams [get_flat_cells u_l2/* -filter "is_hard_macro==true"]
prompt> set l2_mask [create_geo_mask $l2_rams]
prompt> create_placement_attraction -name abc_near_l2 \
 -region [get_attribute $l2_mask bbox] $abc_cells
```

The following ties the cells from the "abc" to be near the cells of the "mmu" module. The location is not restricted to any particular area of the floorplan.

```
prompt> set abc_cells [get_flat_cells u_abc/*]
prompt> set def_cells [get_flat_cells u_mmu/*]
prompt> create_placement_attraction -name abc_near_mmu \
 "$abc_cells $mmu_cells"
```

The following ties cells of the adder module to be close to cells of the multiplier module, and then more loosely ties all cells of the alu together.

```
prompt> set add_cells [get_flat_cells u_alu/u_add/*]
prompt> set mult_cells [get_flat_cells u_alu/u_mult/*]
prompt> set all_alu [get_flat_cells u_alu/*]
prompt> create_placement_attraction -name add_near_mult \
 "$add_cells $mult_cells"
prompt> create_placement_attraction -name all_alu -effort low \
 $all_alu
```

SEE ALSO

remove_placement_attractions(2)
report_placement_attractions(2)
get_placement_attractions(2)
add_to_placement_attraction(2)
remove_from_placement_attraction(2)
create_bound(2)
create_placement(2)

create_placement_blockage

Creates a new placement blockage.

SYNTAX

collection **create_placement_blockage**
-boundary *boundary_spec*
[-type *blockage_type*]
[-purpose user | system]
[-blocked_percentage *percentage*]
[-name *blockage_name*]
[-category *category_name*]
[-cell *cell*]

Data Types

boundary_spec list
blockage_type string
percentage integer
blockage_name string
category_name string
cell collection

ARGUMENTS

-boundary *boundary_spec*

Specifies the boundary of the placement blockage. You can use the following methods to specify the boundary:

- Specify a rectangular boundary by using the following syntax to specify the lower-left and upper-right corners of the rectangle:

  {{*llx lly*} {*urx ury*}}

- Specify a rectilinear boundary by using the following syntax to specify the coordinates of the polygon:

  {{*x1 y1*} {*x2 y2*} ... {*xn yn*}}

- Specify the boundary as a heterogeneous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects.

  In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area includes the areas of each object. In the case of layers, the resulting area includes the area of every shape on the layer.

  When specifying the boundary polygon as a collection of physical objects, the resulting area must resolve to a single, connected polygon. It is an error to specify a collection of objects with a combined area that resolves to multiple polygons.

-type *blockage_type*

Specifies the type of placement blockage to create.

Valid values are

- **hard** (the default)

  A hard blockage prevents the placement of standard cells in the specified region. It is honored by placement, legalization, optimization, and clock tree synthesis.

- **hard_macro**

  A hard macro blockage prevents the placement of hard macros in the specified region. It is honored by placement, legalization, and optimization. This is the only type of placement blockage that is honored by hard macro placement.

- **soft**

  A soft blockage prevents the placement of standard cells in the specified region. It is honored by coarse placement, but not legalization, optimization, and clock tree synthesis.

- **partial**

  A partial blockage limits the cell density in the specified region. It is honored by coarse placement, but not legalization, optimization, and clock tree synthesis.

  When you create a partial blockage, you must specify the blocked percentage value by using the **-blocked_percentage** option.

- **category**

  A category blockage is a partial blockage that also prevents the placement of a specific category of cells in the specified region. It is honored by coarse placement, but not legalization, optimization, or clock tree synthesis.

  The cell categories are identified by a category name attribute. When you create a category blockage, you must specify the category name by using the **-category** option and the blocked percentage value by using the **-blocked_percentage** option.

- **rp_group**

  A relative placement group blockage is a partial blockage that also prevents the placement of relative placement groups in the specified region. It is honored by coarse placement, but not legalization, optimization, or clock tree synthesis.

  When you create a relative placement group blockage, you must specify the blocked percentage value by using the **-blocked_percentage** option.

- **allow_buffer_only**

  An allow-buffer-only blockage allows only buffers and inverters to be placed in the specified region. The cell density is limited to the specified percentage of the blockage area. It is honored by coarse placement, but not legalization, optimization, or clock tree synthesis.

  When you create an allow-buffer-only blockage, you must specify the blocked percentage value by using the **-blocked_percentage** option.

- **allow_rp_only**

  An allow-relative-placement-only blockage allows only relative placement groups to be placed in the specified region; however, hard macros can overlap with the specified region. the cell density is limited to the specified percentage of the blockage area. It is honored by coarse placement, but not legalization, optimization, or clock tree synthesis.

  When you create an allow-relative-placement-only blockage, you must specify the blocked percentage value by using the **-blocked_percentage** option.

- **register**

  A register blockage is a partial blockage that also prevents the placement of registers in the specified region. It is honored by coarse placement, but not legalization, optimization, or clock tree synthesis.

  When you create a register blockage, you must specify the blocked percentage value by using the **-blocked_percentage** option.

-purpose user | **system**

Specifies the purpose of the blockage to be created.

Valid values are:

- **user** (the default)

  This should be used by all user-created blockages and ensures that any automated tools will not remove the blockage.

- **system**

  A system blockage can be removed by tools that create and manipulate blockages. Create a blockage with this purpose only when it can be removed by an automatic tool.

-blocked_percentage ***percentage***

Specifies the percentage blockage for a partial blockage.

This option can only be used with **partial**, **category**, **rp_group**, **allow_buffer_only**, **allow_rp_only**, and **register** blockage types.

-name ***blockage_name***

Specifies the optional name of the blockage. If you specify a name for the blockage, you can get the blockage by name later in the flow.

-category ***category_name***

Specifies the category name for a category blockage.

This option can only be used with the **category** blockage type.

-cell ***cell***

Specifies the physical cell in which to add the placement blockage.

The placement blockage is created in the cell's reference block using the coordinate system of the cell's top-level block. The cell must reference a block, not a library cell, unless this command is executed in the library manager. In the library manager, placement blockages can be created in library cells.

By default, the placement blockage is created in the current block.

DESCRIPTION

This command creates a new placement blockage that is used to control the placement of cells in a specified boundary.

Snapping to the bounding box is done automatically using the global snap settings.

See the description of the **-type** option for more information about the supported placement blockage types and how they are supported by the various engines.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a soft placement blockage.

```
prompt> create_placement_blockage -type soft \
  -boundary {{0 0} {0 100} {100 100} {100 0}}
```

The following example creates a hard placement blockage.

```
prompt> create_placement_blockage -type hard \
  -boundary [get_shapes POLYGON_0]
```

SEE ALSO

create_routing_guide(2)
create_routing_blockage(2)

create_poly_rect

Creates a collection of poly_rect objects.

SYNTAX

collection **create_poly_rect**
-boundary {*boundary_spec*}
[-layers {*layer_list*}]

Data Types

boundary_spec list
layer_list list

ARGUMENTS

-boundary {*boundary_spec*}

A list of rectangles or polygons that specify the boundaries of the poly_rect objects.

Specify each rectangle with the coordinates of its lower-left and upper-right corners: {{*llx lly*} {*urx ury*}}. Specify each polygon with the coordinates of its points: {{*x y*} {*x y*} {*x y*} {*x y*}...}.

This is a required option.

-layers *layer_list*

Specifies a list of layers to associate with each poly_rect in the result. By default, poly_rects are not associated with any layer. This option will associate each poly_rect in the result collection to the respective layer in *layer_list*. To enable this one-to-one correspondence between poly_rects and their respective layers, *layer_list* must have the same length as the list of boundaries.

DESCRIPTION

This command creates a new poly-rectangular object for each rectangle or polygon in the specified list of boundaries, and returns a collection that contains the new objects.

You must specify the boundary of the object, as either a rectangle or a polygon. You may optionally specify a list of layers to associate with each object in the result collection.

EXAMPLES

The following example creates a rectangular poly_rect.

```
prompt> create_poly_rect -boundary {{0 0} {200 100}}
```

The following example creates an L-shaped poly_rect and a rectangular poly_rect.

```
prompt> create_poly_rect -boundary {{{0 0} {0 100} {100 100} \
  {100 200} {200 200} {200 0}} {{0 300} {100 400}}}
```

The following example creates two rectangular poly_rects, and associates the first with layer M1 and the second with layer M2.

```
prompt> create_poly_rect \
  -boundary {{{0 300} {100 400}} {{100 400} {200 500}}} -layers {M1 M2}
```

SEE ALSO

copy_to_layer(2)
create_geo_mask(2)
compute_polygons(2)
resize_polygons(2)
split_polygons(2)
compute_area(2)
transform_polygons(2)

create_port

Creates one or more top-level ports.

SYNTAX

collection **create_port**
[-design *design*]
[-direction in | out | inout]
[-cell *cell*]
[-port_type *port_type*]
port_names

Data Types

cell collection
port_type string
port_names list

ARGUMENTS

-design *design*

Specifies the design in which to create the port(s). If no design is specified, the port(s) are created in the current design.

-direction in | out | inout

Specifies the direction of the ports. By default, the command creates ports with direction **in**.

-port_type *port_type*

Specifies the port_type of the created port. Allowed values are analog_ground, analog_power, analog_signal, clock, deep_nwell, deep_pwell, ground, nwell, power, pwell, reset, scan, signal, tie_high, tie_low and unset. Default port_type is unset which gets treated as signal.

port_names

Specifies the names of the ports to create. Each port name must be unique.

-cell cell

Specifies the cell where the port is to be added. The port is created on the cell's reference module. The cell must not reference a library cell, unless this command is executed in the library manager. In the library manager, ports may be created on library cells. When not specified, the port is created on the current module.

DESCRIPTION

This command creates top-level ports on the current design (unless otherwise specified by -design). It creates only scalar (single-bit) ports. One port is created for each of the names listed. If a port with the specified name already exists, the command displays an error message.

When the **create_port** command creates ports, they are not connected. To establish a connection, you can use the **connect_net** command.

EXAMPLES

The following example creates ports named *data1* and *data2*.

```
prompt> create_port -direction in data1
{"data1"}
prompt> create_port -direction out data2
{"data2"}
```

SEE ALSO

connect_net(2)
disconnect_net(2)
get_ports(2)
remove_ports(2)

create_port_bus

Creates a port bus.

SYNTAX

collection **create_port_bus**
[-design *design*]
[-block *block*]
[-cell *cell*]
[-direction in | out | inout]
[-create_ports]
port_bus_name

Data Types

design collection
block collection
cell collection
port_bus_name string

ARGUMENTS

-design *design*

Specifies the design in which to create the port bus. If no design is specifed, the port bus is created in the current design.

-block *block*

Specifies the block where the port bus is to be added. If specified, **create_port_bus** has the same effect as **edit_module** in which the module is changed and changes are applied to all module occurrences.

-cell *cell*

Specifies the cell where the port bus is to be added. The port bus is created on the cell's reference module. **create_port_bus** first uniquifies the reference if needed, then creates the port bus. When not specified, the port bus is created on the current module.

-direction in | out | inout

Specifies the direction of the ports in port bus. It will not change direction of existing ports. If direction provided in option is not same as existing port direction then port bus will not be created.

-create_ports

Creates port bus members if they do not exist.

port_bus_name

Specifies the port bus name and its range using the **name [start: end]** format. Specify a port bus named busA with port from 1 *to* 5 as **busA[1:5]**. Specify a port bus named busB with port from 5 *downto* as **busB [5:1]**.

DESCRIPTION

This command creates a port bus, collects or creates port bus members, and puts them into the port bus created. This supports creating n-dimensional bus.

The command returns the created port_bus(as a collection), an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

EXAMPLES

The following example creates port bus named *bus1* from port 1 to 5 as bus[1], bus[2], bus[3], bus[4], bus[5]

```
prompt> create_port_bus bus1[1:5]
{"bus1"}
```

The following example first creates ports, then creates a port bus named *bus2* from the created port with indexes 5 downto 1 as bus2[5], bus2[4], bus2[3], bus2[2], bus2[1]

```
prompt> create_port_bus bus2[5:1] -create_ports
{"bus2"}
```

SEE ALSO

add_to_port_bus(2)
get_port_buses(2)
remove_from_port_bus(2)
remove_port_buses(2)
report_port_buses(2)

create_power_domain

Creates or updates a power domain, which provides a power supply distribution network.

SYNTAX

string **create_power_domain**
domain_name
[-elements *list*]
[-exclude_elements *list*]
[-include_scope]
[-supply {*supply_set_handle_name supply_set_name*}*]
[-available_supplies *list*]
[-scope *instance_name*]
[-update]

Data Types

domain_name string
list list
supply_set_handle_name string
supply_set_name string
instance_name string

ARGUMENTS

domain_name

Specifies the name of the power domain to be created. The name should be a simple (non-hierarchical) name.

The power domain cannot be created if there is a power domain with the same name in the specified scope or if there is a hierarchical or leaf cell instance or port with the same name in the specified scope.

An exception to this rule is when you use the **-update** option. In this case the power domain name must be same as an existing power domain in the same scope.

-elements *list*

Specifies a collection of cells (hierarchical or leaf) that are added as an extent of the power domain. Specified cells cannot be added in other power domains of the same scope. The special element **"."** has the same meaning as **-include_scope**, which includes the scope of the power domain in the extent of the domain.

If you do not use either the **-elements** or the **-include_scope** option, the power domain consists of the current scope and any of its children not specified as elements in another **create_power_domain** command.

-exclude_elements *list*

Specifies a collection of cells (hierarchical or leaf) that should not be considered as part of this power domain. Specified cells should also appear in **-elements** list of this power domain and another power domain that the cells should be included.

-include_scope

Includes the scope of the power domain in the extent of the power domain. This means all elements within the current scope get the same supply as the power domain, but they are not explicitly added to it.

-supply {*supply_set_handle_name supply_set_name*}*

Specifies the supply sets that need to be associated with the corresponding supply set handles of the power domain. This option can be specified multiple times.

If the *-supply* is not specified, implicit supply sets are created automatically for each handle of the power domain.

The valid names of the supply set handles are *primary*, *main_rail*, *default_retention* and *default_isolation*.

Specifying the **main_rail** keyword implies that the given domain will be treated as an Always-On Block Synthesis Domain, meaning it will be synthesized with dual-rail cells with primary PG pin connected to the *main_rail* supply and backup PG pin connected to the primary supply of the domain.

Additional handles that can be used apart from the above mentioned ones are **extra_supplies** and **extra_supplies_#**, where **#** represents a unique numeral.

Specifying the **extra_supplies_#** keyword restricts the supply sets that are available for the power domain for usage. The supply sets that are specified with **extra_supplies_#** are made available for the power domain for use in level shifter insertion and always-on synthesis.

Specifying the **extra_supplies** keyword indicates that the power domain does not use any extra supplies.

The **extra_supplies** or **extra_supplies_#** keywords are mutually exclusive with **-update** option.

Using user-defined names for handles is supported.

-available_supplies *list*

Specifies a list of additional supply sets that are available for use in the power domain.

The supply sets that are specified with **-available_supplies** are made available in the power domain for use in level shifter insertion and always-on synthesis. If an empty string argument is specified, only the locally available supplies are available for use by tools to power cells inserted into the power domain.

This option is mutually exclusive with **extra_supplies** or **extra_supplies_#** keywords specified in **-supply** option.

-scope *instance_name*

Specifies in which scope the power domain is to be created. The instance name is the name of a hierarchical cell.

By default, the power domain is created in the current scope.

-update

Instructs the tool to add elements, supply sets and available supplies to an existing power domain. It is an error if the power domain specified does not exist.

If your update is not successful, the tool issues an error message. You can continue to update the functionality, until your update is successful.

DESCRIPTION

This command creates a power domain in the specified scope. A power_domain is a collection of design elements that share a primary power and ground power net. The logic hierarchy level where a power domain is created is called the scope of the power domain. The set of design elements that belong to a power domain are called the "extent" of that power domain. A hierarchical cell is an example of an element. Although a design element can be in the scope of several power domains, it can be in the extent of only one power domain.

A power domain can have several supply nets, which are connected to a power domain via a supply port. A power switch of a power_domain can be used to turn on and off the power supply of part or all of the power domain. You can create a supply net, supply port, and power switch by using the **create_supply_net**, **create_supply_port** and **create_power_switch**, respectively.

When this command succeeds, it returns the full name of the power domain (from the current scope). When it fails, it raises a Tcl error.

EXAMPLES

The following example creates two power domains in the scope named INST1.

```
prompt> create_power_domain PD1 -elements INST1/SUB_INST -scope INST1
INST1/PD1
prompt> create_power_domain PD2 -scope INST1 -include_scope
INST1/PD2
```

The following example creates a power domain with the **-supply** argument using a supply set at the top level scope.

```
prompt> create_power_domain PD_MID -scope mid1 -supply {primary all_primary}
mid1/PD_MID
```

SEE ALSO

create_supply_set(2)
report_power_domains(2)

create_power_state_group

Creates a power state group under current scope.

SYNTAX

string **create_power_state_group**
power_state_group_name

Data Types

power_state_group_name string

ARGUMENTS

power_state_group_name

Specifies the name of the power state group to be created. The name should be a simple (non-hierarchical) name.

The power state group cannot be created if there is a power state group with the same name in the specified scope.

DESCRIPTION

This command creates a power state group in the specified scope. A power_state_group is used to collect related power states defined by add_power_state. The legal power states of a group define the legal combinations power states of other objects in this scope or the descendant subtree. i.e. those combinations of states of those objects that can be active at the same time during operation of the design.

A power_state_group may be used to represent a virtual component made up of more than one instance. Power states defined for the group can represent the legal power states of the virtual component without having to change the design hierarchy to create a single instance for that component.

EXAMPLES

The following example creates a power state group in the scope named SUB1.

```
prompt> set_scope ./SUB1
prompt> create_power_state_group SUB1_GROUP
```

SEE ALSO

add_power_state(2)

create_power_switch

Creates a power switch in the specified power domain.

SYNTAX

string **create_power_switch**
switch_name
-domain *domain_name*
-output_supply_port {*port_name supply_net_name*}
-input_supply_port {*port_name supply_net_name*}
-control_port {*port_name net_name*}
[-ack_port {*port_name net_name* [{*boolean_function*}]}]
[-ack_delay {*port_name delay*}]
-on_state {*state_name input_supply_port* {*boolean_function*}}
[-on_partial_state {*state_name input_supply_port* {*boolean_function*}}]
[-off_state {*state_name* {*boolean_function*}}]
[-supply_set {*supply_set_name*}]
[-error_state {*state_name* {*boolean_function*}}]

Data Types

switch_name string
domain_name string
port_name string
supply_net_name string
net_name string
boolean_function list
delay string
state_name string
input_supply_port string

ARGUMENTS

switch_name

Specifies the name of the power switch to be created. The name should be a simple, non-hierarchical name. If a power switch with the same name already exists in the specified power domain, the power switch cannot be created.

This is a required option.

-domain *domain_name*

Specifies the power domain that contains the power switch. If the power domain with the specified name does not exist in the current scope, the command fails.

This is a required option.

-output_supply_port {*port_name supply_net_name*}

Specifies the name of the output port of the power switch and the supply net to which the port connects. On the power switch, if a port with the same name exists, the command fails. In the current scope, if there is no supply net with the same name, the command fails.

This is a required option.

-input_supply_port {*port_name supply_net_name*}

Specifies the name of the input port of the power switch and the supply net to which the port connects. On the power switch, if a port exists with the same name, the command fails. In the current scope, if there is no supply net with the same name, the command fails.

This is a required option and can be specified more than one time. One power switch can have multiple input supply ports.

-control_port {*port_name net_name*}

Specifies the name of the control port of the power switch and the logical net to which this port connects. If a port with the specified name already exists on the power switch, the command fails. If a net with the specified name does not exist, the command fails.

This is a required option and can be specified more than one time. One power switch can have multiple control ports.

-ack_port {*port_name net_name* {*boolean_function*}}

Specifies the name of the acknowledge port of the power switch and the logical net to which this port connects. If a port with the specified name already exists on the power switch, the command fails. If a net with the specified name does not exist, the command fails.

If this option is not specified, the power switch will not have an acknowledge port. Optionally, you can specify a Boolean function. The *boolean_function* should be defined in terms of the specified control ports, and nothing else.

This option can be specified multiple times.

-ack_delay {*port_name delay*}

Specifies the acknowledge port on the switch and the corresponding acknowledge delay.

This option can be specified multiple times.

-on_state {*state_name input_supply_port* {*boolean_function*}}

Specifies the on-state name, the relevant input supply port, and its Boolean function.

The *boolean_function* should be defined in terms of the specified control ports, and nothing else.

This option can be specified multiple times.

-on_partial_state {*state_name input_supply_port* {*boolean_function*}}

Specifies a named on-partial-state, the relevant input supply port, and its Boolean function.

The *boolean_function* should be defined in terms of the specified control ports, and nothing else.

This option can be specified multiple times.

-off_state {*state_name*, {*boolean_function*}}

Specifies the off-state name and its relevant Boolean function.

The *boolean_function* should be defined in terms of the specified control ports, and nothing else.

This option can be specified multiple times.

-supply_set {*supply_set_name*}

Specifies the supply set that should be used to find the related supply net for control and ack pins. If supply set is not available in the domain, it will be made available if this option is used.

-error_state {*state_name* {*boolean_function*}}

Specifies a named error state and its relevant Boolean function.

The *boolean_function* should be defined in terms of the specified control ports, and nothing else.

This option can be specified multiple times.

DESCRIPTION

The *create_power_switch* command creates a power switch at the specified power domain. The switch is created within the scope of the power domain. Each power switch must be connected to an input supply net and an output supply net. The power switch can be connected with an acknowledge (logical) net and several control (logical) nets. Each net is connected with a power switch via a switch port. The switch ports are automatically created if the power switch is created successfully.

On success, this command returns the full name of the power switch (from the current scope). The command returns a null string upon failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates power switch SW1 within the power domain PD1:

```
prompt> create_power_switch SW1 -domain PD1 \
    -output_supply_port {vout VNO2} \
    -input_supply_port {vin1 VNI1} \
    -input_supply_port {vin2 VNI2} \
    -control_port {ctrl_small ON1} \
    -control_port {ctrl_large ON2} \
    -ack_port {ack_p ACKN {ctrl_small & !ctrl_large}} \
    -ack_delay {ack_p 1} \
    -on_state {full_s vin1 {ctr_small}} \
    -off_state {off_s {!ctr_small}}
```

SEE ALSO

create_power_domain(2)
create_power_switch_array(2)
create_power_switch_ring(2)

create_power_switch_array

Places power-switch cells in the design in an array style. Power switches will be automatically snapped to FinFET grid if FinFET grid exists.

Power switches can be placed before or after compiling the power ground (PG) network. If power switches are placed after generating the PG network, use the **connect_pg_net** and **create_pg_vias** commands after the **create_power_switch_array** command to connect power switch pins to the existing PG network. User can specify -pattern to activate patten flow. It will provide a group of cells placed in a pattern.

SYNTAX

status **create_power_switch_array**
-power_switch *power_switch_strategy*
[-lib_cell *lib_cell*]
[-voltage_area *voltage_area*]
[-voltage_area_shape *voltage_area_shape*]
[-x_pitch *distance*]
[-y_pitch *distance*]
[-x_offset *distance*]
[-y_offset *distance*]
[-siterow_pitch *rowcount*]
[-siterow_offset *rowcount*]
[-prefix *prefix*]
[-orient R0|R90|R180|R270|MX|MY|MXR90|MYR90]
[-checkerboard even | odd]
[-offset_start {*x y*}]
[-pattern *pattern_name*]
[-boundary *polygon*]
[-snap_to_site_row true | false]
[-pg_strategy *pg_strategy_spec*]
[-pg_straps *pg_straps*]
[-align_marker *distance*]
[-switch_number *number*]
[-fine_grid_pitch {*x y*}]
[-search_range {*number number*}]

Data Types

power_switch_strategy string
lib_cell string
voltage_area collection or string
voltage_area_shape collection or string
distance float
rowcount positive integer
prefix string
x float

y float
pattern_name string
polygon list
pg_strategy_spec string
pg_straps collection or string
number integer

ARGUMENTS

-power_switch *power_switch_strategy*

Specifies the power switch strategy name *power_switch_strategy* defined in the UPF file. If **-lib_cell** is not specified, the first library cell that is mapped to the specified power switch is inserted. The **-power_switch** option is not required if the **-lib_cell** option is used. If **-power_switch** is not used, you must ensure that the supply net information is derived correctly for the power switches. This can be achieved by successful power switch associations or by manual associations.

-lib_cell *lib_cell*

Specifies the library cell explicitly. The specified library cell must be mapped to the power switch. This options is required if **-power_switch** is not being used.

-voltage_area *voltage_area*

Specifies the voltage area as the area in which to insert power switches. If not specified, the primary voltage area associated with the power switch is used. The **-voltage_area** option is mutually exclusive with the **-voltage_area_shape** option.

-voltage_area_shape *voltage_area_shape*

Specifies the voltage area shape as the area in which to insert power switches. The **-voltage_area_shape** option is mutually exclusive with the **-voltage_area** option.

-x_pitch *distance*

Specifies the horizontal distance between the centers of two power switches or power switch patterns. If the *distance* is zero, only one column of power switches is inserted. If **-pg_strategy** or **-pg_straps** is specified, power switches are aligned to each vertical power strap when **-x_pitch** is not specified. If **-x_pitch** is also specified, power switches are placed according to pitch, then snapped to the closest power strap.

-y_pitch *distance*

Specifies the vertical distance between centers of two power switches or power switch patterns. If *distance* value is zero, only one power switch row is inserted. **-y_pitch** is mutually exclusive with **-siterow_pitch**.

-x_offset *distance*

Specifies the horizontal offset from the placement region boundary. If *distance* is negative, the tool honors the negative offset, but omits switch cells which would be placed outside the placement boundary.

-y_offset *distance*

Specifies the vertical offset from the placement region boundary. If *distance* is negative, the tool honors the negative offset, but switch cells to be placed outside the placement boundary are dropped. This option is mutually exclusive with **-siterow_offset**.

-siterow_pitch *rowcount*

Specifies the vertical distance, in site rows, between the lower-left corners of two power switches or power switch patterns. **-siterow_pitch** is mutually exclusive with **-y_pitch**.

-siterow_offset *rowcount*

Specifies the vertical offset, in site rows, from the placement region boundary. **-siterow_offset** is mutually exclusive with **-y_offset**.

-prefix *prefix*

Specifies the prefix added to the power switch names. If this option is not specified or the *prefix* is empty, the default prefix *headerfooter* is used.

-orient R0|R90|R180|R270|MX|MY|MXR90|MYR90

Specifies the orientation of the power switch placed at the lower-left corner of the array. The orient can be R0, R90, R180, R270, MX, MY, MXR90, MYR90. If user specified -pattern, it will activate a pattern flow. Only R0 and MX is supported in pattern flow. If not specified, the default orientation is the same as the site row orientation if **-snap_to_site_row** is true. Otherwise, the default orientation is R0.

-checkerboard even | odd

Inserts switch cells in a checkerboard fashion. There are two types of checkerboard: even or odd. By default, the cells are inserted as a normal array.

The even checkerboard type specifies that a switch cell is placed at the lower-left corner. The odd checkerboard type specifies that a blank space is placed at the lower-left corner.

```
x x x x      x x x
 x x x      x x x x
x x x x      x x x
 x x x      x x x x
x x x x      x x x
  even         odd
```

-offset_start {*x y*}

Specifies the startpoint of a uniform power switch grid for disjoint voltage area regions.

-pattern *pattern_name*

Specifies the pattern to be placed. The pattern is defined by the **set_power_switch_placement_pattern** command.

-boundary *polygon*

Specifies the boundary in which to place power switches. The power switches are inserted only in the overlapping area of specified polygon and voltage area or voltage area shape.

-snap_to_site_row true | false

Specifies if the power switches are snapped to site rows. The default value is true.

-pg_strategy *pg_strategy_spec*

Specifies the PG wires with which to align power switches. *pg_strategy_spec* is the strategy name or a triplet that contains the strategy name, net name, and metal layer name. If only the strategy name is specified, power switches are aligned with its input supply net wires on all vertical layers. For example, the **-pg_strategy strategy1** option aligns power switches with both M6 and M8 VDD wires, if the input supply net of the power switch is VDD, and the M6 and M8 VDD wires are generated by strategy1.

If the *pg_strategy_spec* contains the strategy name, net name, and metal layer name, power switches are aligned with those specified wires. For example, the **-pg_strategy {strategy1 VDD M8}** option aligns the power switches with the M8 VDD wires created by using strategy1. Note that the metal layer direction should be vertical. This option is mutually exclusive with **-pg_straps**.

-pg_straps *pg_straps*

Specifies a collection of wires with which to align power switches. You can only specify vertical wires of same PG nets. This option is mutually exclusive with **-pg_strategy**.

-align_marker *distance*

Specifies the marker on the power switch library cell to align with the power straps. The marker is defined as the horizontal offset that refers to the lower-left corner of the library cell. If this option is not specified, the center of the largest rectangular shape of the corresponding pin is the default marker.

-switch_number *number*

Specifies the maximum number of switch cells to be inserted. The switch cell insertion still honors the user-specified offset and pitch. The process stops when the total number of switch cell reach the specified value.

-fine_grid_pitch {*x y*}

Specifies the value to set up a search pitch when power switch cannot be inserted in orginal location. It will search nearby location, it the new location can place power switch successfully. It will stop searching. The option should be used with -search_range.

-search_range {*number number*}

Specifies a pair of integer to set up a search region. The first value is for the number of X-direction pitches. The second is for Y-direction. The option should be used with -fine_grid_pitch.

DESCRIPTION

This command places power-switch cells into the design in an array style. The command also associates the placed cell instances with their power intention as defined in UPF.

Power switch insertion honors placement blockages, keepout margin, macros, fixed cells and other voltage areas as blockages.

EXAMPLES

The following example places power-switch cells in array inside a voltage area.

```
prompt> create_power_switch_array -power_switch Multiplier/mult_sw \
  -x_pitch 10 -y_pitch 10
Insert 684 power switch cell HEAD16DM successfully.
```

SEE ALSO

connect_pg_net(2)
create_pg_vias(2)
create_power_switch(2)
create_power_switch_ring(2)

map_power_switch(2)
report_pg_strategies(2)
report_power_switch_placement_patterns(2)
set_pg_strategy(2)
set_power_switch_placement_pattern(2)

create_power_switch_ring

Places power switch cells around the design in ring style. The power switch cells are automatically snapped to FinFET grid if the FinFET grid exists.

SYNTAX

status **create_power_switch_ring**
-power_switch *power_switch_strategy*
[-lib_cell *lib_cell*]
[-vertical_lib_cell *lib_cell*]
[-inner_corner_cell *lib_cell*]
[-outer_corner_cell *lib_cell*]
[-filler_cells *lib_cell_list*]
[-vertical_filler_cells *lib_cell_list*]
[-voltage_area *voltage_area*]
[-voltage_area_shape *voltage_area_shape*]
[-x_pitch *distance*]
[-y_pitch *distance*]
[-x_offset *distance*]
[-y_offset *distance*]
[-snap_to_site_row true | false]
[-start_point {*x y*}]
[-end_point {*x y*}]
[-boundary *polygon*]
[-through_points *coordinate_list*]
[-prefix *prefix*]
[-orient R0|R90|R180|R270|MX|MY|MXR90|MYR90]
[-vertical_lib_cell_orient R0|R90|R180|R270|MX|MY|MXR90|MYR90]
[-inner_corner_cell_orient R0|R90|R180|R270|MX|MY|MXR90|MYR90]
[-outer_corner_cell_orient R0|R90|R180|R270|MX|MY|MXR90|MYR90]
[-pattern *pattern_name*]
[-continue_pattern]
[-switch_number *integer*]

Data Types

power_switch_strategy string
lib_cell string
lib_cell_list collection or string
voltage_area collection or string
voltage_area_shape collection or string
distance float
x float
y float
polygon list
coordinate_list list
prefix string

pattern_name string
integer integer

ARGUMENTS

-power_switch *power_switch_strategy*

Specifies the power switch strategy name *power_switch_strategy* defined in the UPF file. If **-lib_cell** is not specified, the first library cell that is mapped to the specified power switch is inserted. **-power_switch** is optional as long as **-lib_cell** is used. If not using **-power_switch**, you must ensure that the supply net information is correctly derived for the power switches. This can be achieved by successful power switch associations or by manual associations.

-lib_cell *lib_cell*

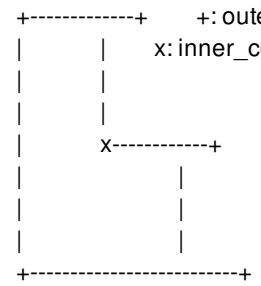
Specifies the library cell explicitly. The specified library cell must be mapped to the power switch. This options is required if **-power_switch** is not being used.

-vertical_lib_cell *lib_cell*

Specifies the library cell to be inserted along vertical edges explicitly. The specified library cell must be mapped to the power switch.

-inner_corner_cell *lib_cell*

Specifies the library cell to be placed at the inner (concave) corners of the region. If not specified, inner corner cell will be the same as cells placed along the edges.



-outer_corner_cell *lib_cell*

Specifies the library cell to be placed at the outer (convex) corners of the region. If not specified, no cell is placed at the outer corners.

-filler_cells *lib_cell_list*

Specifies the list of filler cells to be placed into gaps in the power switch ring. This option can only be used when **-snap_to_site_row** is false. If multiple filler cells are specified, the tool tries to use the filler cells in a descending order of cell size.

-vertical_filler_cells *lib_cell_list*

Specifies the list of filler cells to be placed into gaps in the power switch ring of the vertical edges. This option can only be used when **-snap_to_site_row** is false. If multiple filler cells are specified the tool tries to use the filler cells in a descending order of cell size.

-voltage_area *voltage_area*

Specifies the voltage area as the area in which to insert power switches. If **-voltage_area** is not specified, the primary voltage area associated with the power switch will be used. **-voltage_area** is mutually exclusive with **-voltage_area_shape**, **-**

through_points and **-boundary**.

-voltage_area_shape ***voltage_area_shape***

Specifies the voltage area shape as the area to insert power switches. **-voltage_area_shape** is mutually exclusive with **-voltage_area**, **-through_points** and **-boundary**.

-x_pitch ***distance***

Specifies the horizontal distance between the centers of two power switches or power switch patterns. If *distance* is zero, the pitch set to the width of the switch cell.

-y_pitch ***distance***

Specifies the vertical distance between centers of two power switches or power switch patterns. If *distance* is zero, the pitch is set to the height of the switch cell.

-x_offset ***distance***

Specifies the horizontal offset from the placement region boundary.

-y_offset ***distance***

Specifies the vertical offset from the placement region boundary.

-snap_to_site_row true | **false**

Specifies whether the power switch cells should be snapped to site rows. If **-snap_to_site_row** is true, the cell orientation is flipped when the site row orientation is flipped. If **-snap_to_site_row** is false, cell orientation is rotated along the edges. See the **-orient** option for more information. By default, this option is true.

-start_point {*x y*}

Specifies the startpoint of the partial ring. If the startpoint is not on the edges of the region, it is projected to the closest edge or corner. If the offset value is set, the edges of the expanded or shrunk region are used. This option must be used together with **-end_point**.

-end_point {*x y*}

Specifies the endpoint of the partial ring. If the endpoint is not on the edges of the region, it is projected to the closest edge or corner. If the offset value is set, the edges of the expanded or shrunk region are used. This option must be used together with **-start_point**.

-boundary ***polygon***

Specifies a polygon as the area to insert power switches. The inserted switch cells are associated with the primary voltage area of the power switch strategy. **-boundary** is mutually exclusive with **-voltage_area**, **-voltage_area_shape** and **-through_points**.

-through_points ***coordinate_list***

Specifies a list of points as the edges to insert power switches. The inserted switch cells are associated with the primary voltage area of the power switch strategy. **-through_points** is mutually exclusive with **-voltage_area**, **-voltage_area_shape** and **-boundary**.

-prefix ***prefix***

Specifies the prefix added to power switch names. If this option is not specified or the specified string is empty, the default prefix *headerfooter* is used.

-orient R0|R90|R180|R270|MX|MY|MXR90|MYR90

Specifies the orientation of the power switch placed along the bottom-most and left-most edge. Legal orientation values are: R0,

R90, R180, R270, MX, MY, MXR90, and MYR90. The default orientation is R0. Cell orientation along other edges are rotated accordingly. If **-snap_to_site_row** is true, cell orientation is flipped when site row orientation is flipped. If **-snap_to_site_row** is false, the rotation rules are: O_RIGHT = R90(O_BOTTOM); O_TOP = MX(O_BOTTOM); O_LEFT = MXR90(O_BOTTOM). O_BOTTOM, O_RIGHT, O_LEFT, O_TOP are cell orientations along bottom, right, left, and top edges, respectively.

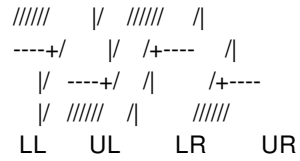
-vertical_lib_cell_orient R0|R90|R180|R270|MX|MY|MXR90|MYR90

Specifies the orientation of the power switch placed along the right-most edges. Legal orientation values are: R0, R90, R180, R270, MX, MY, MXR90, and MYR90. The default orientation is R0. Cell orientation along other edges are rotated accordingly. If **-snap_to_site_row** is true, cell orientation is flipped when site row orientation is flipped. If **-snap_to_site_row** is false, the rotation rules are: O_LEFT = MY(O_RIGHT). O_RIGHT and O_LEFT are cell orientations along right and left edges, respectively.

-inner_corner_cell_orient R0|R90|R180|R270|MX|MY|MXR90|MYR90

Specifies the orientation of power switch placed at the lower-left corner. Legal orientation values are: R0, R90, R180, R270, MX, MY, MXR90, and MYR90. The default orientation is R0. The orientation of other corners are rotated accordingly. The rules are: O_UL = MX(O_LL); O_LR = MY(O_LL); O_UR = R180(O_LL). O_LL, O_UL, O_LR, and O_UR are the orientations of LL, UL, LR, UR corners, respectively.

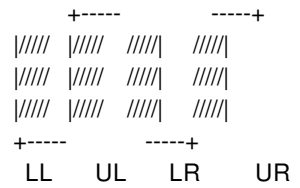
The inner corner types are:



-outer_corner_cell_orient R0|R90|R180|R270|MX|MY|MXR90|MYR90

Specifies the orientation of power switch placed at the LL corner. Legal orientation values are: R0, R90, R180, R270, MX, MY, MXR90, and MYR90. The default orientation is R0. The orientations of other corners are rotated accordingly. The rules are: O_UL = MX(O_LL); O_LR = MY(O_LL); O_UR = R180(O_LL). O_LL, O_UL, O_LR, and O_UR are the orientations of LL, UL, LR, UR corners, respectively.

The outer corner types are:



-pattern *pattern_name*

Specifies the pattern name to be inserted. The pattern is defined by the **set_power_switch_placement_pattern** command. When **-pattern** is specified, the **-x_pitch** and **-y_pitch** options are ignored. This option can only be used when **-snap_to_site_row** is false.

-continue_pattern

Specifies that the pattern should restart on adjacent edges. By default, a pattern restarts on the next adjacent edge. If **-continue_pattern** is specified, the current pattern continues. This option is ignored if **-pattern** is not specified.

-switch_number *integer*

Specifies the total number of power switches to be inserted. If offset and pitch is specified, the insertion will follow the pitch and stop when the number of inserted cells reaches the specified value. If **-pitch** is not specified, the tool derives the pitch to evenly distribute the required number of switch cells along the ring.

DESCRIPTION

This command places power switch cells into a power switch ring. The command also associates the placed cell instances with their power intention as defined in UPF.

Power switch insertion honors placement blockages, keepout margin, macros, fixed cells and other voltage areas as blockages.

EXAMPLES

The following example places power switch cells along the boundary of the prime voltage area of the power switch strategy.

```
prompt> create_power_switch_ring -power_switch Multiplier/mult_sw \
  -x_pitch 10 -y_pitch 10

Insert 64 power switch cell HEAD16DM successfully.
```

SEE ALSO

create_power_switch(2)
create_power_switch_array(2)
map_power_switch(2)
report_power_switch_placement_patterns(2)
set_power_switch_placement_pattern(2)

create_pr_rule

Creates a cell row spacing rule (pr_rule) in the specified technology object.

SYNTAX

collection **create_pr_rule**
[-tech *tech_object*]
[-library *library*]
-row_spacing *distance_list*
-abut_table *abut_table*

Data Types

tech_object collection
library collection
distance_list list
abut_table list

ARGUMENTS

-tech *tech_object*

Specifies the technology object in which to create the cell row spacing rule.

The **-tech** and **-library** options are mutually exclusive; you can specify only one. If you do not specify either option, the rule is created in the technology object of the current library.

-library *library*

Specifies the library in which to create the cell row spacing rule. The command creates the rule in the technology object associated with the specified library. In an implementation tool, the library is a design library. In the library manager, the library is a cell library. You can specify the library using the library's name or a collection that contains the library.

The **-tech** and **-library** options are mutually exclusive; you can specify only one. If you do not specify either option, the rule is created in the technology object of the current library.

-row_spacing *distance_list*

Specifies the row spacing distances. Use the following syntax to specify the distances, where each value is a floating point number that represents the spacing in microns:

{*top_top top_bottom bottom_bottom*}

- The *top_top* argument represents the spacing between the top edges when you are using double-back cell rows in your floorplan.

- The *top_bottom* argument represents the spacing between the top edge and the bottom edge when you are not using double-back cell rows in your floorplan.
- The *bottom_bottom* argument represents the spacing between the bottom edges when you are using double-back cell rows in your floorplan.

-abut_table *abut_table*

Specifies whether the edges can be abutted. Use the following syntax to specify the *abut_table* argument, where each value is either **true** (allows abutment) or **false** (does not allow abutment):

{*top_top top_bottom bottom_bottom*}

- The *top_top* argument specifies whether the two top edges can be abutted when you are using double-back cell rows in your floorplan.
- The *top_bottom* argument specifies whether the top edge and the bottom edge can be abutted when you are not using double-back cell rows in your floorplan.
- The *bottom_bottom* specifies whether the two bottom edges can be abutted when you are using double-back cell rows in your floorplan.

DESCRIPTION

This command creates a new cell row spacing rule (pr_rule) in the specified technology object. The tool treats the rule the same as a rule defined in the PRRule section of the technology file. For more information about defining cell spacing rules, see the "PRRule Section" in the *Synopsys Technology File and Routing Rules Reference Manual*.

If the command is successful, it returns a collection that contains the created rule; otherwise, it returns an empty string. If there is a command syntax error, it returns a TCL_ERROR.

EXAMPLES

The following example creates a cell row spacing rule in the technology object of the current library.

```
prompt> create_pr_rule -row_spacing {1.0 1.1 1.2} \
  -abut_table {false true false}
```

SEE ALSO

get_pr_rules(2)
remove_pr_rules(2)
report_pr_rules(2)

create_pst

Creates a UPF power state table (PST), assigns a name to the table, and lists the supply ports or nets in a specific order.

SYNTAX

string **create_pst**
table_name
-supplies *list*

Data Types

table_name string
list list

ARGUMENTS

table_name

Specifies the name of the power state table.

-supplies *list*

Specifies a list of supply nets or supply ports to include in the power state table. Hierarchical names are allowed.

DESCRIPTION

The **create_pst** command creates a new UPF power state table, assigns a name to the table, and lists the supply ports or supply nets in a specific order. After you use this command, use the **add_port_state** command to specify the names and voltages of the possible states for each supply port, and the **add_pst_state** command to specify the allowed combinations of supply states.

The following script demonstrates the power state definition flow:

```
# Create power state table, specify supply ports for table
create_pst MyPST -supplies { PN1 PN2 SOC/PN3 }

# Specify supply states and corresponding voltages
add_port_state PN1     -state {sLO 0.88}
add_port_state PN2     -state {sLO 0.88} -state {sHI 0.99}
add_port_state SOC/PN3 -state {sLO 0.88} -state {PDOWN off}

# Specify power states (allowed combinations of supply states)
```

```
add_pst_state S1 -pst MyPST -state {sLO sLO sLO}
add_pst_state S2 -pst MyPST -state {sLO sLO PDOWN}
add_pst_state S3 -pst MyPST -state {sLO sHI PDOWN}
```

The power state table defines the legal combinations of states that can exist at any given time during the operation of the design. The supply states sLO, sHI, and PDOWN represent the low voltage, high voltage, and power-down states of the supplies. The power states S1, S2, and S3 each refer to a specific combination of supply states for the supplies listed in the **create_pst** command.

The **create_pst** command can only be used at the top-level scope. Power switch supply ports are considered supply ports because they are connected by supply nets, so they can be listed as supply nets in the **create_pst** command.

A supply port and a supply net can have the same name, even when they are unconnected. If such a name is listed in the **create_pst** command, it is assumed to represent the supply port and not the supply net.

The **create_pst** command returns the name of the power state table if the table is created successfully, or the null string otherwise.

You can use multiple power state tables to specify the states within a particular scope. The tool expands the specified tables to cover all possible combinations of states.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command creates a power state table named MyPST and specifies a list of three supplies to be included in the table:

```
prompt> create_pst MyPST -supplies {PN1 PN2 SOC/PN3}
```

SEE ALSO

add_port_state(2)
add_pst_state(2)
report_pst(2)

create_purpose

Creates a purpose in the given tech object.

SYNTAX

collection **create_purpose**
[-tech *tech_object*]
-name *purpose_name*
-number *purpose_number*
[-force]

Data Types

tech_object collection
purpose_name string
purpose_number int

ARGUMENTS

-tech *tech_object*

Purpose will be created in this tech object. If not specified then tech object of current library will be used.

-name *purpose_name*

Specifies the name of the purpose to be created.

-number *purpose_number*

Specifies the number of the purpose to be created.

-force

If specified, it overwrites an existing purpose of the same name.

DESCRIPTION

create_purpose command creates a purpose in the given technology. The purpose creation requires a name and a number for the new purpose.

If the command is successful, it returns the newly-created purpose as a single-element collection. If the command fails, it returns TCL_ERROR with appropriate error message. If there is a syntax error, it returns a TCL_ERROR.

EXAMPLE

The following example creates a purpose in the technology of the current design.

```
prompt> create_purpose -tech [get_techs [current_lib]] -name test_purpose -number 100
{test_purpose}
```

The following example creates the same purpose without -tech option.

```
prompt> create_purpose -name test_purpose -number 100
{test_purpose}
```

SEE ALSO

get_techs(2)
remove_tech(2)

create_qor_snapshot

Generates a quality-of-results report for specified modes/corners/scenarios and stores the report into a set of files under location designated by application option **time.snapshot_storage_location**.

SYNTAX

status **create_qor_snapshot**
-name *name*
[-power]
[-significant_digits *digits*]
[-zero_wire_load]
[-max_paths *number*]
[-nworst *number*]
[-scenarios *list_of_scenarios*]
[-nosplit]

Data Types

name string
digits integer
number integer
list_of_scenarios list

ARGUMENTS

-name *name*

Specifies the name of the QoR snapshot, which can be used to identify a particular snapshot. The name should start with a letter.

-power

Records dynamic and static power.

-significant_digits *digits*

Specifies the number of significant digits for the values generated in the reports.

-zero_wire_load

Uses zero wire load models for timing reporting.

-max_paths *number*

Specifies the maximum number of paths for the **report_timing** command to report per path group.

-nworst *number*

Specifies the maximum number of worst paths for the **report_timing** command to report per timing endpoint.

-scenarios *list_of_scenarios*

Specifies the list of active scenarios to get a QoR snapshot. The default behavior is to report all active scenarios.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **create_qor_snapshot** command generates a quality-of-results report for given scenarios of the current design and stores the report into a set of files. A QoR snapshot contains information about quality metrics such as timing, area, DRC, clocks, power.

Multicorner-Multimode Support

By default, this command works on all active scenarios. To specify different scenarios, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following command creates a QoR snapshot named "preroute" from the current design and saves the snapshot files under the "snapshot" directory under the current working directory.

```
prompt> create_qor_snapshot -name preroute
```

The following example shows how the number of significant digits can affect the output of the Timing.

```
prompt> create_qor_snapshot -name atest -significant_digits 4
 ...
**************************************************
No. of scenario = 1
s1 = default
---------------------------------------------------
WNS of each timing group:                  s1
---------------------------------------------------
Clk                              -0.4855
---------------------------------------------------
Setup WNS:                        -0.4855
Setup TNS:                      -141.7032
Number of setup violations:               450
Hold WNS:                            -
Hold TNS:                            -
Number of hold violations:                -
Number of max trans violations:           0
Number of max cap violations:             0
Number of min pulse width violations:          0
---------------------------------------------------
```

```
Area:                         16574.746
Cell count:                        2607
Buf/inv cell count:                 758
Std cell utilization:              0.7444
CPU(s):                            7
Mem(Mb):                           231
Host name:                         host
----------------------------------------------------

prompt> create_qor_snapshot -name atest -significant_digits 3
 ...
****************************************************
No. of scenario = 1
s1 = default
----------------------------------------------------
WNS of each timing group:                s1
----------------------------------------------------
Clk                              -0.486
----------------------------------------------------
Setup WNS:                       -0.486
Setup TNS:                     -141.703
Number of setup violations:            450
Hold WNS:                           -
Hold TNS:                           -
Number of hold violations:             -
Number of max trans violations:        0
Number of max cap violations:          0
Number of min pulse width violations:      0
----------------------------------------------------
Area:                         16574.746
Cell count:                        2607
Buf/inv cell count:                 758
Std cell utilization:              0.744
CPU(s):                            7
Mem(Mb):                           231
Host name:                         host
----------------------------------------------------
```

SEE ALSO

query_qor_snapshot(2)
remove_qor_snapshot(2)
report_qor_snapshot(2)
time.snapshot_storage_location(3)

create_rdl_power_extension

Creates RDL power extensions.

SYNTAX

status **create_rdl_power_extension**
-layer *tech_layer_name*
[-mode new | remove]
[-power_net *net_name*]
[-ground_net *net_name*]
[-bbox { {*llx lly*} {*urx ury*} }]
[-width_variation { *variation distance*}]
[-remove_nets *nets*]
[-from_previous_extension true | false]

Data Types

tech_layer_name string
net_name string
llx float
lly float
urx float
ury float
variation float
distance float
nets collection

ARGUMENTS

-layer *tech_layer_name*

Specifies the metal layer on which to optimize the RDL wires. You specify the metal layer by using its technology file layer name. This option is required.

Use the **create_routing_rule** command to specify the required spacing of the thin nets before using this command.

-mode new | remove

Specifies the action to perform: add new extensions or remove existing extensions. The **-mode new** option creates new RDL extensions for power and ground straps. If a net was previously extended, even partially, no new RDL extensions are created for that net.

The **-mode remove** option removes existing RDL extensions. This option can remove only those RDL extensions that were created with the **create_rdl_power_extension** command. Wires created by other means, such as a third-party tool, might not have full association information and must be removed by using the **remove_routes** command. If the tool cannot find any RDL

extensions, no action is performed.

-power_net ***net_name***

Specifies which power net to extend wires. By default, the command automatically finds power net to extend wires. If the design contains multiple power nets, you must use this option to specify the power net.

-ground_net ***net_name***

Specifies which ground net to extend wires. By default, the command automatically finds ground net to extend wires. If the design contains multiple ground nets, you must use this option to specify the ground net.

-bbox { {*llx lly*} {*urx ury*} }

Specifies the bounding box of a rectangular routing area. The engine will only extend power and ground nets inside the area. The format {{*llx lly*} {*urx ury*}} specifies the lower-left and upper-right corners of the rectangle.

-width_variation { *variation distance*}

Specifies the width variation of the extension wires. Inside each distance, the difference of wire widths from small to large (the direction is from startpoint to endpoint) will be equal or smaller than the variation. If the variation is zero, it means the wire width will become equal or smaller only. The unit for variation and distance is microns.

-remove_nets ***collection_of_nets***

Specifies a list of net names to remove extension.

-from_previous_extension true | false

Create extension from previous extension (default false).

DESCRIPTION

This command extends power straps from the core area to the I/O region. The routing widths of the RDL extensions are made as wide as possible while still meeting DRC rules. Before running this command, you must first create power straps. The command searches for starting points from power straps that are within a search distance to the bumps. The search distance is the average distance between the flip chip bumps. To efficiently use available RDL routing space, you should run the **optimize_rdl_routes -reserve_power_resources true** command to reserve resources. You can use the **create_routing_rule** command to change the internal application of DRC rules. The maximum wire width can be controlled by the maxWidth value setting in the technology file.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates RDL power extensions on the METAL9 layer.

```
prompt> create_rdl_power_extension -layer METAL9
```

The following example first reserves resources, then creates RDL power extensions on the METAL7 layer.

```
prompt> optimize_rdl_routes -reserve_power_resources true -layer METAL7
```

```
prompt> create_rdl_power_extension -layer METAL9
```

The following example removes the RDL power extensions created by the **create_rdl_power_extension** command.

```
prompt> create_rdl_power_extension -mode remove
```

SEE ALSO

create_pg_strap(2)
create_routing_rule(2)
optimize_rdl_routes(2)

create_rdl_routing_guides

Creates routing guides for RDL routing at block corners to honor special spacing rules.

SYNTAX

collection **create_rdl_routing_guides**
-layer *layer*
-check_overlap_layer *layer*
-corner_distance *distance*
-boundary_distance *distance*
[-spacing *distance*]
[-name_prefix *name_prefix*]

Data Types

layer list
distance double
name_prefix string

ARGUMENTS

-layer *layer*

Specifies the layer on which to set the rdl routing guide.

-check_overlap_layer *layer*

Specifies the layer to check whether there are shapes in the corner regions. Only if there are shapes in the corner regions on the check_overlap_layer, then the rdl routing guide at these corner regions are created.

-corner_distance *distance*

Specifies the distance from the block corner. This distance is the length of two sides of the pentagonal routing guides.

-boundary_distance *distance*

Specifies the distance from the block boundary. This distance is the length of two sides of the pentagonal routing guides.

-spacing *distance*

Specifies the spacing for RDL router to honor inside the routing guides. The **spacing** value must be larger than the **minSpacing** specified in the tech file. If this option is not specified, use the value of **nonDefaultRDLRegionMinSpacing** defined in the technology file as spacing.

-name_prefix *name_prefix*

Specifies the prefix of the name of the created routing guides.

DESCRIPTION

This command creates several new rdl type routing guides and returns a collection that contains the created routing guide.

First, the command removes all existing rdl type routing guides. Then, it checks the pentagon regions at four boundary corner to see if there is any geometry on the **-check_overlap_layer** in these regions. The pentagon regions are calculated by the values of **-corner_distance** and **-boundary_distance**. Then, it creates rdl type routing_guides only in the corner regions which has overlapped geometries on the **-check_overlap_layer**. Finally, the command associates the routing guides with the **nonDefaultRDLRegionMinSpacing** value (defined in technology file) as the special spacing. If some corner regions do not contain geometries on the **-check_overlap_layer**, no rdl type routing guides are created for those corners. You can also use **-spacing** to specify the spacing value for the rdl routing guides.

The lower-left corner pentagon region is calculated as follows:

If the boundary bounding box of the block is *(ll_x, ll_y) (ur_x, ur_y)*, and the **-corner_distance** is *c_dist*, **-boundary_distance** is *b_dist*, then the boundary of the lower-left corner pentagon region is

(ll_x, ll_y) (ll_x + c_dist, ll_y) (ll_x + c_dist, ll_y + b_dist) (ll_x + b_dist, ll_y + c_dist) (ll_x, ll_y + c_dist)

The other three pentagon regions are calculated symmetrically in the corresponding corners.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates rdl routing guides on layer M8. Two rdl routing guides are created, since geometries on AP layer only exist at these two block corners.

```
prompt> create_rdl_routing_guides -layer {M8} -check_overlap_layer AP -boundary_distance 600 -corner_distance 750 -sp
```

```
Information: Remove existing RDL routing guide before creating new one. (RT-305)
{RDL_0 RDL_1}
```

SEE ALSO

create_routing_guide(2)

create_rdl_shields

Performs automatic shield routing on RDL nets.

SYNTAX

status **create_rdl_shields**
-layers *layer_list*
[-mode new | unshield | reshield | tie | report]
[-nets *collection_of_nets* | -nets_in_file *nets_file*]
[-shield_on_bump true | false]
[-shield_via_tie true | false]
[-shield_routing_tie true | false]
[-trim_floating true | false]
[-reference_layer layer]
[-offset offset_value]
[-report_coaxial_shield_ratio_offsets coaxial_shield_layer_offset_list]
[-coordinates bounding_box_list]
[-half_shield off | up | down | left | right]

Data Types

layer_list collection
collection_of_nets collection
nets_file string
offset_value unsigned int
bounding_box_list collection
coaxial_shield_layer_offset_list collection

ARGUMENTS

-layers *layer_list*

Specifies the target layers for creating shielding wires. Use layer names from the technology file.

-mode new | unshield | reshield | tie | report

Selects the action to perform during automatic shielding.

The mode options are defined as follows:

- **new** (default)
  Creates new shielding wires for unshielded RDL nets. If a net is previously shielded, even partially, no new shielding wires are created for that net.

You can specify the power or ground net shielding wires for a net with the **set_app_options {flip_chip.route.shielding_net shielding_net}** command before you run the **create_rdl_shields** command. If the shielding net is unset, the RDL router uses the ground net with maximum number of ports as the shielding net.

- **unshield**
  Removes existing shielding wires that are associated with shielded RDL nets. The command removes all tie-off connections from deleted shielding wires to the power and ground network.

  This command can remove only those shielding wires that were created by using the **create_rdl_shields** command. Shielding wires created by other means, such as a third-party tool, might not have full association information and must be removed with a different command.

  If the tool cannot find any shielding wires associated with shielded RDL nets, no action is performed.

- **reshield**
  Removes existing RDL net shields and creates new shielding wires. The shielding wires are first removed as described in the **-mode unshield**. After removing all existing shielding wires, new shielding wires are added for the specified RDL nets that do not have shielding wires.

  If the design does not contain existing shielding wires, it is more efficient to use **-mode new**, because the tool first checks for existing shielding WIRES and then creates new shielding WIRES.

- **tie**
  Do "tie shielding wires" operation only.

- **report**
  Do "report shield ratio" operation only.

-nets *collection_of_nets*

Specifies the RDL nets to be shielded, unshielded, or reshielded, depending on the specified mode.

-nets_in_file *file_name*

Specifies the name of the file that contains the RDL nets to be shielded, unshielded, or reshielded, depending on the specified mode.

-nets has higher priority than **-nets_in_file**.

If neither **-nets** nor **-nets_in_file** option is specified, the tool processes all RDL nets.

When creating shielding wires, the tool skips specified RDL nets, which have no shielding rules defined in **create_routing_rule**, or the **-shield_widths** defined for the target layer is 0. To define the shielding rules, use the **create_routing_rule** and **set_routing_rule** commands.

-shield_on_bump true | false

Specifies whether to create shielding wires around bump cells. The default is true.

-shield_via_tie true | false

Specifies whether to tie shielding wires with vias. The default is true.

-shield_routing_tie true | false

Specifies whether to tie shielding wires with RDL routes. The default is true.

-trim_floating true | false

Specifies whether to remove floating shielding wires. The default is true.

-reference_layer layer

Specifies the layer of routes which shields are created for. It is designed for coaxial shield.

When -reference_layer is specified, only one layer is allowed in -layers. Coaxial shields are created on the layer specified in -layers.

-offset offset_value

Specifies the offset of coaxial shields to their referenced routes. Default is 0.

- **offset_value = 0, shields are created to align to center line of target routes.**
- **offset_value > 0, shields are created by shifting center line of target routes by offset_value**
- **offset_value < 0, invalid value.**

-report_coaxial_shield_ratio_offsets {{shield_layer1 route_layer1 offset1} {shield_layer2 route_layer2 offset2} ...}

By default, only side wall shield ratio is reported after create_rdl_shields command. With this option specified, the coaxial shield ratio will be also reported. Users can specify a list of layer offset triple {shield_layer route_layer shield_route_offset}. These tripe values are used to determine the layer of shields to report, the layer of route to be shielded, and the effective shield to route distance

-coordinates {bounding_box1 bounding_box2 ..}

Specifies the area that create_rdl_shields command applies. Default value is whole design boundary.

create_rdl_shields creates shields and tie these shields to PG structure. When running on a huge design, for example, an interpose design, it might have performance issue when tie-ing shields due the large number of PG shapes. Users could use this option to improve the run time of create_rdl_shields.

-half_shield {off | up | down| left | right}

Specifies whether to create side-wall shield on one side.

By default, it is off, and create_rdl_shields create side wall shields on two sides.

- **For horizontal nets, create upside side-wall shield when -half_shield is "up", and crate downside side-wall shield when -half_shield is "down".**
- **For vertical nets, create left side-wall shield when -half_shield is "left", and create right side-wall shield when -half_shield is "right".**

DESCRIPTION

The **create_rdl_shields** command shields RDL nets based on the associated shielding rules. Shielding rules are defined by the **create_routing_rule** command. The shielding wires are tied to RDL wires, standard pins, and standard cell rails of shielding net, defined by the **set_app_options {flip_chip.route.shielding_net shielding_net}** command. If no shielding net is defined, the RDL router uses the ground net with maximum number of ports as the shielding net.

RDL nets are nets in flip-chip or 3DIC designs which are routed on the redistribution layer. In a flip-chip design, RDL nets connect a bump cell to driver I/O cells. In a 3DIC or interposer design, RDL nets can connect the following elements:

- A microbump cell and a microbump cell
- A microbump cell and a through silicon via (TSV) cell
- A microbump cell and driver I/O cell

Microbump cells are also called flip-chip pads.

Prerequisites

Before you run this command, you must first define the shielding rules with the **create_routing_rule** command and assign these rules to the RDL nets to be shielded with the **set_net_routing_rule** command. You might need to set a specific power or ground net that shielding wires belong to, by using **set_app_options {flip_chip.route.shielding_net shielding_net}** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines the shielding rules and assigns the shielding rules to CLK_B1 and CLK_B5. It also assigns VSS as shielding net, and routes the CLK_B1 and CLK_B5 nets. Finally, it shields the CLK_B1 and CLK_B5 nets with the VSS ground net, and uses via and RDL routes to tie shielding wires to VSS without trimming any floating shielding wires.

```
prompt> create_routing_rule clock_net_shielding \
  -widths { METAL7 10 METAL8 10 } \
  -spacings { METAL7 2 METAL8 2 } \
  -shield \
  -shield_widths { METAL7 10 METAL8 10 } \
  -shield_spacings { METAL7 2 METAL8 2 } \

prompt> set_routing_rule -rule clock_net_shielding {CLK_B1 CLK_B5}
prompt> set_app_options -block [current_block] {flip_chip.route.shielding_net VSS}
prompt> route_rdl_flip_chip -layers METAL8
prompt> create_rdl_shields -layers METAL8 -nets {CLK_B1 CLK_B5} \
      -shield_routing_tie true -shield_via_tie true -trim_floating false
```

The following example creates METAL7 coaxial shielding wires to shield METAL8 routings. The shield to route centerline offset is 5.

```
prompt> create_rdl_shields -reference_layer METAL8 -layers METAL7 -offset 5\
      -report_coaxial_shield_ratio {{METAL7 METAL8 5}}
```

The following example creates METAL7 shielding wires within the area {100 100} {200 200}.

```
prompt> create_rdl_shields -layers METAL7 -coordinates {{100 100} {200 200}}
```

SEE ALSO

create_routing_rule(2)
remove_routes(2)
set_routing_rule(2)

create_repeater_groups

Groups a list of repeaters or repeaters of supernets.

SYNTAX

status **create_repeater_groups**
-supernets *supernet_list* | -supernet_bundles *supernet_bundle_list*
| -cells *cell_list*
-lib_cells *lib_cell_list*
[-lib_cell_input *lib_pin*]
[-lib_cell_output *lib_pin*]
[-group_pin_spacing *spacing*]
[-ignore_pin_layers]

Data Types

supernet_list	list
supernet_bundle_list	list
cell_list	list
lib_cell_list	list
lib_pin	string
spacing	float

ARGUMENTS

-supernets *supernet_list*

Specifies a list of supernets whose repeaters are grouped. The list might contain supernet names, patterns, or collections. Specify a collection by using the **get_supernets** command.

-supernet_bundles *supernet_bundle_list*

Specifies a list of supernet bundles whose repeaters are grouped. The list might contain supernet bundle names, patterns, or collections. Specify a collection by using the **get_bundles** command. When you use this option, you cannot use the **-group_pin_spacing** and **-ignore_pin_layers** options.

-cells *cell_list*

Specifies a list of cells for grouping. The list might contain cell names, patterns, or collections. Specify a collection by using the **get_cells** command.

-lib_cells *lib_cell_list*

Specifies a list of library cells that enables you to group repeaters whose library cells is in the list specified by the -lib_cells

option. The list might contain library cell names, patterns, or collections. Specify a collection by using the **get_lib_cells** command.

-lib_cell_input ***lib_pin***

Specifies the input pin name of the library cell. By default, the command uses "D" as the input pin name.

-lib_cell_output ***lib_pin***

Specifies the output pin name of the library cell. By default, the command uses "Q" as the output pin name.

-group_pin_spacing ***spacing***

Specifies the distance within which two pins are considered to be in one group. The default is 2.0 um.

-ignore_pin_layers

Pin layers are ignored when grouping pins. With this option, pins in the same group might not be on the same layer.

DESCRIPTION

This command automatically groups a list of repeaters or repeaters of supernets. This command passes in a list of repeaters using **-cells**, a list of supernets using **-supernets** or bundles of supernets using **-supernet_bundles** and return a list of repeater group created by **set_repeater_group**.

To know the number of newly created repeater groups and their group IDs use Tcl variables, crg_num_of_created_group_ids and crg_created_group_ids. To verify each repeater group, use the **report_repeater_groups** command.

-supernets, **-supernet_bundles** and **-cells** are mutual exclusive. When you use **-supernet_bundles**, you are not allowed to set **-group_pin_layers** and **-ignore_pin_layers**.

The command supports both single load and multiple loads supernets.

EXAMPLES

The following examples shows usages of different repeater inputs separately.

prompt> **create_repeater_groups -supernets [get_supernets sNet*] -lib_cells {lib_cell_1 lib_cell_2}**

prompt> **create_repeater_groups -supernet_bundles [get_bundles sBundle*] -lib_cells [get_lib_cells libCell*]**

prompt> **create_repeater_groups -cells [get_cells eCell*] -lib_cells lib_cell_1**

The following example uses the **-group_pin_spacing** option to group pins within a specific distance.

prompt> **create_repeater_groups -supernets [get_supernets] -lib_cells {lib_cell_1 lib_cell_2} -group_pin_spacing 5.0**

The following example uses the **-ignore_pin_layers** option to ignore layer difference of pins.

prompt> **create_repeater_groups -cells [get_cells eCell*] -lib_cells {lib_cell_1 lib_cell_2} -ignore_pin_layers**

SEE ALSO

set_repeater_group(2)
report_repeater_groups(2)
place_group_repeaters(2)

create_repelling_group_bound_shapes

Create and associate routing blockages from repelling group bound content.

SYNTAX

string **create_repelling_group_bound_shapes** [-guard_band *distance*] [-name_prefix *routing_blockage_name_prefix*]
-repelling_group_bounds *repelling_group_bounds*

list *distance*
string *routing_blockage_name_prefix*
list *repelling_group_bounds*

ARGUMENTS

-guard_band *distance*

Distance value(s) for guard band in form of {x,y} or {r}. The computed cover shapes are based on a core's internal cells, enlarged by the distance provided here. Additionally routing blockages are associated with a core's internal nets. These routing blockages prevent the internal nets from being routed over hostile cores in the repelling group bound.

-name_prefix *routing_blockage_name_prefix*

Optional name prefix for the routing blockages created by this command.

-repelling_group_bounds *repelling_group_bounds*

This option specifies the repelling group_bounds for which routing blockages are created.

DESCRIPTION

This command produces several routing blockages for each core found in the given repelling group bounds. Blockages are created on all routing layers. Two types of routing blockages are created per core.

The first type of routing blockages represent the cover shape of a core, and their blockage_group_id is associated with the core by the core's attribute called rgb_core_cover_shape. Although these blockages are not explicitly associated with any nets, their shape determines whether internal nets or transit nets are protruding.

The second type of routing blockages confine the routing space of a core's internal nets, and their blockage_group_id is associated with the internal nets and also with the core (by attribute rgb_core_routing_blockages). Their shape for a given core is the union of:

- the complement of this core's cover shape

- the cover shape of the hostile cores in the repelling group bound.

EXAMPLES

The following example creates a repelling group bound containing two cores called dcls1 and dcls2, and based on the current legalized placement, obtains a list of shapes per core.

```
prompt> create_bound -name "MyRGB1" -repelling diamond -dimensions 10 [get_cells {dcls1 dcls2}]
prompt> create_placement
prompt> legalize_placement
prompt> create_repelling_group_bound_shapes -repelling_group_bounds [get_bounds MyRGB] -guard_band 6

Information: Created routing blockage RB_60 (blockage_group_id 1, layer M1). (SR-001)
Information: Created routing blockage RB_61 (blockage_group_id 1, layer M2). (SR-001)
Information: Created routing blockage RB_62 (blockage_group_id 1, layer M3). (SR-001)
Information: Created routing blockage RB_63 (blockage_group_id 1, layer M4). (SR-001)
Information: Created routing blockage RB_71 (blockage_group_id 2, layer M1). (SR-001)
Information: Created routing blockage RB_72 (blockage_group_id 2, layer M2). (SR-001)
Information: Created routing blockage RB_73 (blockage_group_id 2, layer M3). (SR-001)
Information: Created routing blockage RB_74 (blockage_group_id 2, layer M4). (SR-001)
Information: Created routing blockage RB_82 (blockage_group_id 3, layer M1). (SR-001)
Information: Created routing blockage RB_83 (blockage_group_id 3, layer M2). (SR-001)
Information: Created routing blockage RB_84 (blockage_group_id 3, layer M3). (SR-001)
Information: Created routing blockage RB_85 (blockage_group_id 3, layer M4). (SR-001)
Information: Created routing blockage RB_93 (blockage_group_id 4, layer M1). (SR-001)
Information: Created routing blockage RB_94 (blockage_group_id 4, layer M2). (SR-001)
Information: Created routing blockage RB_95 (blockage_group_id 4, layer M3). (SR-001)
Information: Created routing blockage RB_96 (blockage_group_id 4, layer M4). (SR-001)
```

SEE ALSO

create_bounds
report_safety_status

create_routing_blockage

Creates a routing blockage on metal, via, or poly layers.

SYNTAX

collection **create_routing_blockage**
-layers *layers*
-boundary *boundary_spec*
[-net_types *net_type_list*]
[-zero_spacing]
[-reserve_for_top_level_routing]
[-boundary_internal]
[-boundary_external]
[-allow_metal_fill_only]
[-cell *cell*]
[-blockage_group_id *group_id_number*]
[-name_prefix *prefix*]
[-allow_via_ladder]

Data Types

layers collection
boundary_spec list
net_type_list list
cell collection
group_id_number integer
prefix string

ARGUMENTS

-layers *layers*

Specifies the layers on which to create the routing blockage. Specify the layers by using a collection or by specifying a list of layer names from the technology file.

If you specify more than one layer, the tool creates multiple routing blockages simultaneously.

This is a required option.

-net_types *net_type_list*

Specifies the net types that must respect the generated routing blockages.

You can specify one or more of the following values: **analog_ground**, **analog_power**, **analog_signal**, **clock**, **deep_nwell**, **deep_pwell**, **ground**, **nwell**, **power**, **pwell**, **reset**, **scan**, **signal**, **tie_high**, and **tie_low**.

The default is all net types.

This option is mutually exclusive with the **-blockage_group_id**, **-boundary_internal**, and **-boundary_external** options.

-boundary *boundary_spec*

Specifies the boundary of the routing blockage. The boundary can be a rectangle or a polygon.

To specify a rectangle, use the following format to specify its lower-left and upper-right coordinates:

{ {*llx lly*} {*urx ury*} }

To specify a polygon, use the following format to specify its coordinates:

{ {*x1 y1*} {*x2 y2*} ... {*xn yn*} }

You can also specify a polygon as the combined area of a heterogeneous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area includes the areas of each object. In the case of layers, the resulting area includes the area of every shape on the layer.

When specifying the boundary polygon as a collection of physical objects, the resulting area must resolve to a single, connected polygon. It is an error to specify a collection of objects with a combined area that resolves to multiple polygons.

This is a required option.

-zero_spacing

Creates a zero-spacing routing blockage.

A zero-spacing routing blockage disables the minimum spacing rule between the routing blockage boundary and the net shapes. When you use this option, the net shapes can touch, but not overlap, the routing blockage boundary.

This option is mutually exclusive with the **-boundary_internal** and **-boundary_external** options.

-reserve_for_top_level_routing

Creates a routing blockage to reserve area for top-level routing.

Top-level routing uses the area reserved by this kind of routing blockages as a routing channel through the block.

This option is mutually exclusive with the **-boundary_internal**, **-boundary_external** and **-allow_metal_fill_only** options.

-boundary_internal

Creates an internal boundary routing blockage.

This option is mutually exclusive with the **-reserve_for_top_level_routing**, **-boundary_external**, and **-allow_metal_fill_only** options.

-boundary_external

Creates an external boundary routing blockage.

This option is mutually exclusive with the **-reserve_for_top_level_routing**, **-boundary_internal**, and **-allow_metal_fill_only** options.

-allow_metal_fill_only

Creates a routing blockage that allows only metal fill shapes within the routing blockage. No routing shapes are allowed within the blockage.

You must use the **-layers** option with this option to specify the layer on which to create the routing blockage.

This option is mutually exclusive with the **-boundary_internal**, **-boundary_internal**, and **-reserve_for_top_level_routing** options.

-cell ***cell***

Specifies the physical cell in which to add the routing blockage.

The command creates the routing blockage in the cell's reference block using the coordinate system of the cell's top-level block.

The cell must reference a block, not a library cell, unless you run the command in the library manager. In the library manager, you can create routing blockages in library cells.

By default, the routing blockage is created in the current block.

-blockage_group_id ***group_id_number***

Assigns a blockage group identification number to the routing_blockage created by the command. You can assign the same blockage group ID number to Multiple blockages by using multiple **create_routing_blockage** commands. All routing blockages with the same blockage group ID belong to the same group. For block boundary blockages, the group ID number is an integer in the range of 0 to 99. For other routing blockages, the group ID number is an integer in the range from 1 to 99 and defaults to 0 if unspecified.

Each net has an integer attribute named **routing_blockage_group_id**, which lets you specify which blockage group applies to that net. For example, if a net's **routing_blockage_group_id** attribute is set to 3, all routing blockages assigned a blockage group ID of 3 block routing of that net in the layers specified for the routing blockages.

This option is mutually exclusive with the **-net_types** option.

-name_prefix ***prefix***

Specifies a prefix string used for the names of the generated routing blockage.

When you use this option, the tool uses the following naming convention for the generated routing blockages:

prefix_object_id

If you do not use this option, the tool uses the following naming convention for the generated routing blockages:

RB_*object_id*

-allow_via_ladder

Allows via ladders to overlap the routing blockage.

DESCRIPTION

The **create_routing_blockage** command creates metal or via routing blockages on the metal, via, or poly layers. The routing blockages can be either rectangular or rectilinear.

Unlike routing guides, which direct the router to change routing information on nets that go through the routing guide, routing blockages direct the router to avoid routing through these areas.

EXAMPLES

The following example creates a rectangular via routing blockage on the VIA5 layer with its lower-left corner at (0,0) and its upper-right corner at (100,100).

```
prompt> create_routing_blockage -layers VIA5 \
  -boundary {{0 0} {100 100}}
{RB_29440}
```

The following example creates rectilinear metal routing blockages on the M1 and VIA4 layers with boundary points at (0,0), (50,0), (50,50), (100,50), (100,100), and (0,100).

```
prompt> create_routing_blockage -layers {M1 VIA4} \
  -boundary {{0 0} {50 0} {50 50} {100 50} {100 100} {0 100}}
{RB_29696 RB_29697}
```

The following example creates a rectangular metal routing blockage with its lower-left corner at (0,0) and its upper-right corner at (50,50) on each metal layer in the design library:

```
prompt> create_routing_blockage \
  -layers [get_layers  -filter {name=~M*}] \
  -boundary {{0 0} {50 50}}
{RB_29441 RB_29442 RB_29443 RB_29444 RB_29445 RB_29446 RB_29447
RB_29448 RB_29449 RB_29450 RB_29451 RB_29452 RB_29453 RB_29454 RB_29455}
```

The following two commands create two routing blockages belonging to the same routing blockage group with group ID number 3:

```
prompt> set rBlockage1 [create_routing_blockage -layers M3 \
  -boundary {{0 0} {100 100}} -blockage_group_id 3]
{RB_29500}
prompt> set rBlockage2 [create_routing_blockage -layers VIA4 \
  -boundary {{0 0} {100 100}} -blockage_group_id 3]
{RB_29501}
```

The following command specifies group ID number 3 as the routing blockage group associated with net n52, causing blockages rBlockage1 and rBlockage2 to block routing of n52 in the specified blockage layers:

```
prompt> set_attribute -name routing_blockage_group_id \
  -objects [get_nets n52] -value 3
{n52}
```

The following command returns the routing blockages that belong to group ID number 3:

```
prompt> get_routing_blockages -filter "blockage_group_id==3"
{RB_29500 RB_29501}
```

SEE ALSO

get_layers(2)
get_routing_blockages(2)
remove_routing_blockages(2)

create_routing_corridor

Creates a new routing corridor.

SYNTAX

collection **create_routing_corridor**
[-boundary { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects]
[-path {{x y} {x y} ... }]
[-width width]
[-start_endcap flush | half_width | full_width]
[-end_endcap flush | half_width | full_width]a
[-min_layer_name min_layer]
[-max_layer_name max_layer]
[-name corridor_name]
[-object objects]
[-cell cell]

Data Types

llx	float
lly	float
urx	float
ury	float
x	float
y	float
geometric_objects	collection
width	float
min_layer	collection
max_layer	collection
corridor_name	string
objects	collection
cell	string or collection

ARGUMENTS

-boundary { {{*llx lly*} {*urx ury*}} | {{*x y*} {*x y*} {*x y*} {*x y*}...} } | geometric_objects

Specifies the boundary coordinates of the routing corridor. The boundary may be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates (i.e., {*llx lly*} {*urx ury*}). A polygon is specified by its points (i.e., {*x y*} {*x y*} {*x y*} {*x y*}...).

Polygons may also be specified as the combined area of a heterogenous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

When specifying the boundary polygon as a collection of physical objects, the resulting area must resolve to a single, connected polygon. It is an error to specify a collection of objects with a combined area that resolves to multiple polygons.

The **-boundary** and **-path** options are mutually exclusive, you must specify only one.

-path {{*x y*} {*x y*} ... }

Specifies the coordinates for a rectilinear path shape to be added to the routing corridor. A path must have 2 or more points. The **-width** option is required when you specify the **-path** option. The **-boundary** and **-path** options are mutually exclusive, you must specify only one.

-width *width*

Specifies the width of each segment of the path specified for a path shape. The **-path** option is required when you specify the **-width** option.

-start_endcap flush | half_width | full_width

Specifies the endcap for the startpoint of the corridor path. The endcap values are flush, half_width, and full_width. The default is flush.

-end_endcap flush | half_width | full_width

Specifies the endcap for the endpoint of the corridor path. The possible values are flush, half_width, and full width. The default is flush.

-min_layer_name *min_layer*

Specifies the minimum routing layer for the routing corridor shape. Only one layer can be specified. By default, the minimum layer is the min routing layer.

-max_layer_name *max_layer*

Specifies the maximum routing layer for the routing corridor shape. Only one layer can be specified. By default, the maximum layer is the max routing layer.

-name *corridor_name*

Specifies the name of the routing corridor. By default, the tool generates a unique name for the routing corridor by using the format CORRIDOR_objId.

-object *objects*

Specifies the nets and supernets (individual or bundled) to associate with the routing corridor.

-cell *cell*

Specifies the physical cell where the routing corridor is to be added. The routing corridor is created in the cell's reference block using the coordinate system of the cell argument's top block. The cell must reference a block, not a library cell, unless this command is executed in the library manager. In the library manager, routing corridors can be created in library cells. When not specified, the routing corridor is created in the current block.

DESCRIPTION

This command creates a new routing corridor based on the boundary or path you specify. You can associate nets and supernets in the design with a routing corridor of the same design only and restrict their routing to within the routing corridor.

EXAMPLES

The following example creates a routing corridor with one rectangle shape, and associates six nets with the corridor.

```
prompt> create_routing_corridor -name corridor_a \
  -boundary { {35 35} {37 45} } \
  -min_layer_name M1 -max_layer_name M4 \
  -object [get_nets [list N1 N2 N3 N4 N5 N6]]
```

The following example creates a routing corridor with one rectilinear shape, and associates one net and one supernet with the corridor.

```
prompt> create_routing_corridor -name corridor_b \
  -boundary {{35 35} {45 35} {45 40} {40 40} {40 45} {35 45} } \
  -min_layer_name M1 -max_layer_name M4 \
  -object {N3 supernet_1}
```

The following example creates a routing corridor with one path shape, and associates two nets with the corridor.

```
prompt> create_routing_corridor -name corridor_c \
  -path {{35 35} {45 35} {45 40} } -width 0.2 \
  -min_layer_name M1 -max_layer_name M4 \
  -object [get_nets [list N5 N6]]
```

SEE ALSO

add_to_routing_corridor(2)
create_routing_corridor_shape(2)
get_routing_corridor_shapes(2)
get_routing_corridors(2)
remove_from_routing_corridor(2)
remove_routing_corridor_shapes(2)
remove_routing_corridors(2)
report_routing_corridors(2)
routing_corridor_attributes(3)
routing_corridor_shape_attributes(3)

create_routing_corridor_shape

Creates a routing corridor shape on a routing corridor.

SYNTAX

collection **create_routing_corridor_shape**
-routing_corridor *routing_corridor*
[-boundary { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects]
[-path {{*x y*} {*x y*} ... }]
[-width *width*]
[-start_endcap flush | half_width | full_width]
[-end_endcap flush | half_width | full_width]
[-min_layer_name *min_layer*]
[-max_layer_name *max_layer*]

Data Types

routing_corridor collection
llx float
lly float
urx float
ury float
x float
y float
geometric_objects collection
width float
min_layer collection
max_layer collection

ARGUMENTS

-routing_corridor *routing_corridor*

Specifies the routing corridor to create the routing corridor shape. This is a required option.

-boundary { {{*llx lly*} {*urx ury*}} | {{*x y*} {*x y*} {*x y*} {*x y*}...} } | geometric_objects

Specifies the boundary coordinates of the routing corridor shape. The boundary may be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates (i.e., {*llx lly*} {*urx ury*}). A polygon is specified by its points (i.e., {*x y*} {*x y*} {*x y*} {*x y*}...).

Polygons may also be specified as the combined area of a heterogeneous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other

physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

When specifying the boundary polygon as a collection of physical objects, the resulting area must integrate to a single, connected polygon. It is an error to specify a collection of objects with a combined area that integrates into multiple polygons.

The **-boundary** and **-path** options are mutually exclusive, you must specify only one.

-path {{*x y*} {*x y*} ... }

Specifies the coordinates for a rectilinear path shape to be added to the routing corridor. A path must have 2 or more points. The **-width** option is required when you specify the **-path** option. The **-boundary** and **-path** options are mutually exclusive, you must specify only one.

-width *width*

Specifies the width of each segment of the path specified for a path shape. The **-path** option is required when you specify the **-width** option.

-start_endcap flush | half_width | full_width

Specifies the endcap for the starting point of the corridor path. The endcap values are flush, half_width, and full_width. The default is flush.

-end_endcap flush | half_width | full_width

Specifies the endcap for the end point of the corridor path. The possible values are flush, half_width, and full width. The default is flush.

-min_layer_name *min_layer*

Specifies the minimum routing layer for the routing corridor shape. Only one layer can be specified.

-max_layer_name *max_layer*

Specifies the maximum routing layer for the routing corridor shape. Only one layer can be specified.

DESCRIPTION

This command creates a routing corridor shape and adds it to the specified routing corridor. The names of the routing corridor shapes are automatically generated.

The **-boundary** and **-path** options are mutually exclusive, you must specify only one of these options.

EXAMPLES

The following example adds a rectangle routing corridor shape to the routing corridor named CORRIDOR_1.

```
prompt> create_routing_corridor_shape -routing_corridor CORRIDOR_1 \
  -boundary { {35 35} {37 45} } \
  -min_layer_name M1 -max_layer_name M4
{"CORRIDOR_SHAPE_2"}
```

The following example adds a rectilinear routing corridor shape to the routing corridor named CORRIDOR_1.

```
prompt> create_routing_corridor_shape -routing_corridor CORRIDOR_1 \
  -boundary {{35 35} {45 35} {45 40} {40 40} {40 45} {35 45} }  \
  -min_layer_name M1 -max_layer_name M4
{"CORRIDOR_SHAPE_3"}
```

The following example adds a path routing corridor shape to the routing corridor named CORRIDOR_1.

```
prompt> create_routing_corridor_shape -routing_corridor CORRIDOR_1 \
  -path {{35 35} {45 35} {45 40} } -width 0.2 \
  -min_layer_name M1 -max_layer_name M4
{"CORRIDOR_SHAPE_4"}
```

SEE ALSO

add_to_routing_corridor(2)
create_routing_corridor(2)
get_routing_corridor_shapes(2)
remove_from_routing_corridor(2)
remove_routing_corridor_shapes(2)
remove_routing_corridors(2)
report_routing_corridors(2)
report_routing_corridors(2)
routing_corridor_attributes(3)
routing_corridor_shape_attributes(3)

create_routing_guide

Creates a new routing guide.

SYNTAX

collection **create_routing_guide**
-boundary *list_of_points*
[-layers *layers*]
[-horizontal_track_utilization *percentage*]
[-vertical_track_utilization *percentage*]
[-name *guide_name*]
[-cell *cell*]
[-river_routing]
[-switched_direction_only]
[-max_patterns *pattern_string*]
[-access_preference *access_preference*]
[-metal_cut_allowed]
[-forbidden_preferred_grid_extension]
[-design_boundary_blockage]
[-pin_access]
[-standard_cell_region]
[-single_row_via_ladder_pattern_must_join_allowed]
[-rdl_routing *spacing*]
[-preferred_direction_only]
[-switch_preferred_direction]

Data Types

list_of_points integer
layers list
percentage integer
guide_name string
cell collection
pattern_string string
access_preference string
spacing double

ARGUMENTS

-boundary *list_of_points*

Specifies the coordinates of the bounding box of the routing guide. The coordinate unit is specified in the technology file.

This option is required.

-layers *layers*

Specifies the layers on which to set the routing guide.

This option is required with the **-river_routing**, **-switched_direction_only**, and **-max_patterns** options. This option is not required when using other options.

The direction of the track utilization is determined from the preferred direction of the specified layers. If the preferred direction of a specified layer is horizontal, the horizontal track utilization is set on that layer. If you specify more than one utilization in the same area, the router uses the minimum utilization.

-horizontal_track_utilization *percentage*

Specifies the horizontal track utilization for the routing guide.

You must specify an integer value between 0 and 100 for the *percentage* argument. If you do not specify this option or if you specify a value greater than 100, it is set to 100.

Note that a utilization routing guide is a soft constraint for the router. It does not impact the capacity analysis in global routing. Also, for the overlapping area of multiple routing guides, the lowest utilization value of the routing guides is honored in the overlapping area.

-vertical_track_utilization *percentage*

Specifies the vertical track utilization for the routing guide.

You must specify an integer value between 0 and 100 for the *percentage* argument. If you do not specify this option or if you specify a value greater than 100, it is set to 100.

Note that a utilization routing guide is a soft constraint for the router. It does not impact the capacity analysis in global routing. Also, for the overlapping area of multiple routing guides, the lowest utilization value of the routing guides is honored in the overlapping area.

-name *guide_name*

Specifies the name of the created routing guide.

If you do not specify this option, the tool assigns a name to the routing guide using the naming convention RG#x, where x is an integer value.

-cell *cell*

Specifies the physical cell where the routing guide needs to be added. When you use this command in an implementation tool, the cell must reference a block, not a library cell. When you run this command in the library manager, the cell can reference a library cell.

The routing guide is created in the cell's reference block using the coordinate system of the cell argument's top block.

If you do not specify this option, the routing guide is created in the current block.

-river_routing

Specifies that the type of route guide is **river_routing**.

River routing refers to the routing of many nonshorting routes in roughly the same direction when routing is already restricted to just one or two layers.

When you use this option, you must also use the **-layers** option to specify the layers on which river routing is encouraged.

This option is mutually exclusive to **-switched_direction_only**, **-max_patterns**, and **-access_preference** options.

-switched_direction_only

Specifies that the type of route guide is **switched_direction_only**.

When you use this option, you must also use the **-layers** option to specify the affected layers.

This option is mutually exclusive to **-river_routing**, **-max_patterns**, and **-access_preference** options.

-max_patterns ***pattern_string***

Specifies that the type of route guide is **max_patterns**.

When you use this option, you must also use the **-layers** option to specify the affected layers.

This option is mutually exclusive to **-river_routing**,**-switched_direction_only**, and**-access_preference** options.

-access_preference ***access_preference***

Specifies that the type of route guide is **access_preference**

Use the following syntax to specify the access preference areas for the routing guide:

{*layer* {*access_preference_type* { {*x1 y1*} {*x2 y2*} } *strength*}}

Where *access_preference_type* is either **wire_access_preference** or **via_access_preference**, *x1* and *y1* are the coordinates of the lower-left corner of the access preference area, *x2* and *y2* are the upper-right coordinates of the access preference area and *strength* is a value between 0 and 1. If access preference areas overlap, the stronger access preference area takes precedence over the weaker access preference area. To require routing in a specific access preference area, use a strength value of 1. When you define access preference areas with a strength of 1, all access preference areas with a strength less than 1 are ignored and Zroute treats the access preference routing guide as a hard constraint.

This option is mutually exclusive to **-river_routing**, **-switched_direction_only**, and **-max_patterns** options.

-metal_cut_allowed

Specifies that the type of route guide is **metal_cut_allowed**.

-forbidden_preferred_grid_extension

Specifies that the type of route guide is **forbidden_preferred_grid_extension**.

-design_boundary_blockage

Specifies that the type of route guide is **design_boundary_blockage**.

-pin_access

Specifies that the type of route guide is **pin_access**.

-standard_cell_region

Specifies that the type of route guide is **standard_cell_region**.

-single_row_via_ladder_pattern_must_join_allowed

Specifies that the type of route guide is **single_row_via_ladder_pattern_must_join_allowed**.

-rdl_routing ***spacing***

Specifies that the type of route guide is **rdl_routing**, with the spacing as specified. The **spacing** value must be larger than the **minSpacing** specified in the tech file.

-preferred_direction_only

Creates a preferred-direction-only routing guide.

If you use the **-layers** option, the setting applies only to the specified layers; otherwise, it applies to all layers within the route guide boundary.

-switch_preferred_direction

Creates a switch-preferred-direction routing guide.

If you use the **-layers** option, the setting applies only to the specified layers; otherwise, it applies to all layers within the route guide boundary.

DESCRIPTION

This command creates a new routing guide and returns a collection that contains the created routing guide.

The tool uses the global snap settings to snap the bounding box of the routing guide.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a routing guide.

```
prompt> create_routing_guide -boundary {{0.0 0.0} {100.0 100.0}} \
    -layers {M3 M4} \
    -river_routing \
    -horizontal_track_utilization 50 -vertical_track_utilization 30
```

SEE ALSO

get_routing_guides(2)
remove_routing_guides(2)
create_routing_blockage(2)
get_routing_blockages(2)
remove_routing_blockages(2)

create_routing_rule

Defines non-default routing rules in the design.

SYNTAX

status **create_routing_rule**
[-default_reference_rule | -reference_rule_name *ref_rule_name* |
-no_reference_rule]
[-widths {*layer_name width* ...}]
[-shield]
[-shield_widths {*layer_name width* ...}]
[-shield_spacings {*layer_name spacing* ...}]
[-snap_to_track]
[-vias {{*via_definition_name site_spec rotation_spec*} ...} |
-cuts {{*cut_layer_name* {*cut_name num_of_cuts*} ...} ...}]
[-taper_distance *distance*]
[-driver_taper_distance *distance*]
[-taper_over_pin_layers *num_of_layers*]
[-taper_under_pin_layers *num_of_layers*]
[-driver_taper_over_pin_layers *num_of_layers*]
[-driver_taper_under_pin_layers *num_of_layers*]
[-spacings {*layer_name* {*spacing* ...} ...}]
[-spacing_weight_levels {*layer_name* {*[low|medium|high|hard* ...} ...}]
[-spacing_length_thresholds {*layer_name* {*spacing_length_threshold* ...} ...}]
[-multiplier_width *width_multiplier*]
[-multiplier_spacing *spacing_multiplier*]
[-single_side_spacing]
[-ignore_spacing_to_pg true|false]
[-ignore_spacing_to_blockage true|false]
[-ignore_spacing_to_shield true|false]
[-single_connection_to_pin true|false]
[-rdl_taper_distances {*layer_name distance* ...}]
[-rdl_taper_widths {*layer_name width* ...}]
[-mask_constraints {{*layer_name mask_name* [fixed]} ...}]
[-via_spacings {{*layer_name layer_name spacing*} ...}]
[-parallel_wire {layer_name ...}]
rule_name

Data Types

cut_layer_name	string
cut_name	string
distance	float
layer_name	string
mask_name	string
num_of_cuts	integer
num_of_layers	integer

ref_rule_name string
rule_name string
spacing float
spacing_multiplier float
width float
width_multiplier float
rule_name string

ARGUMENTS

-default_reference_rule

Specifies that the default routing rule (from the tech) is used as the reference rule when initializing the width and shielding constraints. See the **-widths** and **-shield** options.

The **-default_reference_rule**, **-reference_rule_name**, and **-no_reference_rule** options are mutually exclusive; you may use only one. By default, the command uses the default routing rule as the reference rule when none of these options are specified.

-reference_rule_name *ref_rule_name*

Specifies the name of the reference (source) rule to use. The created routing rule is initialized with the values from the reference rule.

The **-default_reference_rule**, **-reference_rule_name**, and **-no_reference_rule** options are mutually exclusive; you may use only one. By default, the command uses the default routing rule as the reference rule when none of these options are specified.

-no_reference_rule

Specifies that no reference rule should be used. The created routing rule will not be initialized with values from any reference rule. See the **-widths** and **-shield** options.

The **-default_reference_rule**, **-reference_rule_name**, and **-no_reference_rule** options are mutually exclusive; you may use only one. By default, the command uses the default routing rule as the reference rule when none of these options are specified.

-widths {*layer_name width* ...}

Specifies the routing width for each named routing layer. Each entry in the list is comprised of a routing layer name and its width separated by a space. The width portion of the pair is a floating point number. You can specify only a single width per layer.

If you do not specify the width for a layer, the tool copies the width from the reference rule if a reference rule is specified. See the **-default_reference_rule**, **-reference_rule_name**, and **-no_reference_rule** options.

-shield

Defines shielding in the non-default routing rule using the default spacing and width. No shielding constraints are defined in the routing rule if the **-no_reference_rule** option is specified.

-shield_widths {*layer_name width* ...}

Specifies the shielding width for each named routing layer. Each entry in the list is comprised of a routing layer name and its shielding width separated by a space. The shielding width portion of the pair is a floating point number. You can specify only a single width per layer.

If you do not specify the shielding width for a layer, the tool copies the shielding width from the reference rule specified by **-reference_rule_name**. If **-reference_rule_name** is not specified, then the shielding width defaults to unspecified for the layer.

-shield_spacings {*layer_name spacing* ...}

Specifies the minimum shield spacing allowed between wires for each named routing layer. Each entry in the list is comprised of a routing layer name and its shield spacing separated by a space. The shield spacing portion of the pair is a floating point number. You can specify only a single spacing per layer.

If you do not specify the shield spacing for a layer, the tool copies the shield spacing from the reference rule specified by **-reference_rule_name**. If **-reference_rule_name** is not specified, then the shield spacing defaults to unspecified for the layer.

-snap_to_track

Snaps shielding wires to the nearest routing track.

-vias {{*via_definition_name site_spec rotation_spec*} ...}

Creates vias automatically during routing, using this via pattern. Each entry in the via_pattern_list argument consists of the routable via definition name, its horizontal and vertical site count numbers, and its rotation setting, separated by spaces. The via definition name can be found as one of the simple via_def names. To obtain a list of simple via_defs, call the following command:

```
get_via_defs -filter "via_def_type==simple_via_def"
```

The via_defs are generated from the Technology File's ContactCode, DEF VIAS definitions, and the **create_via_def** command. The site count portion of the triplet is in the format mxn, where m is the horizontal site count (i.e., the number of columns in the via) and n is the vertical site count (i.e., the number of rows in the via), separated by the x character. The rotation portion of the triplet is specified by r or nr, where r indicates a rotated via and nr indicates an unrotated via. The rotation portion of the triplet is specified by r or nr, where r indicates a rotated via and nr indicates an unrotated via. Only those modes and rotations that are explicitly specified are allowed by the non-default routing rule. For example, to allow all via array modes (swapped and unswapped) and rotations (rotated and unrotated) for a 1x2 via array named VIAHH, you must specify the following definition:

```
-vias {{VIAHH 1x2 NR} {VIAHH 1x2 R} {VIAHH 2x1 NR} {VIAHH 2x1 R}}
```

When you define a via array, e.g. {VIAHH 2x2 R}, the 2x2 via array is constructed by rotating VIAHH instances. The rotation portion of the triplets should be specified on the cuts of the via array, not on the via array itself.

The **-vias** and **-cuts** options are mutually exclusive; you can specify only one.

-cuts {{*cut_layer_name* {*cut_name num_of_cuts*} ...} ...}

Creates vias automatically during routing using the general cut definitions.

Each entry in the option value consists of the cut layer name and the cut pair lists, separated by spaces. Each cut pair consists of the cut name and the number of cuts, separated by spaces. The cut name is defined in the Layer section of the technology file along with the width and height information for the cut. The **create_routing_rule** command searches for proper {via, rowXcolumn, rotation} solutions (the description for the **-via** option for more information) without violating any non-default rules or technology file rules.

The solutions must satisfy the following conditions:

1. The cut sizes of the vias match the width and height of the specified cut name.
2. The solutions are a subset of the contact codes defined in the fatTblFatContactNumber attribute in the Layer section of the technology file required by the metal width and number of cuts.
3. The cut number is the larger of the number defined in the non-default routing rule and the number defined in the fatTblFatContactMinCuts attribute in the Layer section of the technology file.

For example, assume the following information is defined in the technology file:

```
Layer "VIA1" {
  fatTblThreshold        = ( 0, 0.181, 0.411 )
  fatTblFatContactNumber  = ( "2,3,4,5,6", "5,6,20", "5,6,20" )
  fatTblFatContactMinCuts = ( "1,1,1,1,1", "1,1,1",  "2,2,2" )
```

```
    cutNameTbl          = ( Vsq, Vrect )
    cutWidthTbl         = ( 0.05, 0.05 )
    cutHeightTbl        = ( 0.05, 0.13 )
    ...
  }
```

To specify all cuts that match the size requirement of Vrect, meet the fat table rules according to the upper and lower metal widths, and the cut number is at least 1, enter the following definition:

```
create_routing_rule ruleA -widths { M1 0.2 M2 0.25 } -cuts {{VIA1 {Vrect 1}}}
```

The **create_routing_rule** command automatically populates all cuts that meet the requirements.

The **-vias** and **-cuts** options are mutually exclusive; you can specify only one.

-taper_distance *distance*

Specifies the taper distance for all routing layers. If you do not specify the taper distance, the tool copies the taper distance from the reference rule specified by **-reference_rule_name**. If **-reference_rule_name** is not specified, then the taper distance defaults to unspecified, and Signal router will use 10 times the mean number of tracks as the tapering distance for all routing layers and the default tapering wire width is the default width for the respective layer.

-driver_taper_distance *distance*

Specifies the driver taper distance for all routing layers. If you do not specify the driver taper distance, the tool copies the driver taper distance from the reference rule specified by **-reference_rule_name**. If **-reference_rule_name** is not specified, the the driver taper distance defaults to unspecified, and Signal router will use 10 times the mean number of tracks as the driver tapering distance for all routing layers and the default tapering wire width is the default width for the respective layer.

-taper_over_pin_layers *num_of_layers*

Specifies the number of routing layers over the pin layer that the router may use for tapering, and enables layer-based pin tapering.

Values must be greater or equal to 0. If both the **-taper_over_pin_layers** and **-taper_under_pin_layers** options are set to zero, pin tapering is disabled on the pin layer. If either option is set to a nonzero value, the pin layer is considered a taper layer.

If this option is not specified for this rule, the tool copies it from the reference rule specified by **-reference_rule_name**. If **-reference_rule_name** is not specified, then the taper over pin layers defaults to unspecified, and the router will apply tapering up to 2 layers over the pin layer. The default is 2.

If any of the layer-based tapering options is specified, distance-based tapering options (i.e. **-taper_distance** and **-driver_taper_distance**) are ignored.

-taper_under_pin_layers *num_of_layers*

Specifies the number of routing layers under the pin layer that the router may use for tapering, and enables layer-based pin tapering.

Values must be greater or equal to 0. If both the **-taper_over_pin_layers** and **-taper_under_pin_layers** options are set to zero, pin tapering is disabled on the pin layer. If either option is set to a nonzero value, the pin layer is considered a taper layer.

If this option is not specified for this rule, the tool copies it from the reference rule specified by **-reference_rule_name**. If **-reference_rule_name** is not specified, then the taper under pin layers defaults to unspecified, and the router will apply tapering down to 2 layers under the pin layer. The default is 2.

If any of the layer-based tapering options is specified, distance-based tapering options (i.e. -taper_distance and -driver_taper_distance) are ignored.

-driver_taper_over_pin_layers num_of_layers *num_of_layers*

Specifies the number of routing layers over the output pin layer that the router may use for tapering, and enables layer-based pin tapering.

Values must be greater or equal to 0. If both the **-driver_taper_over_pin_layers** and **-driver_taper_under_pin_layers** options are set to zero, pin tapering is disabled on the pin layer. If either option is set to a nonzero value, the pin layer is considered a taper layer.

If this option is not specified for this rule, the tool copies it from the reference rule specified by **-reference_rule_name**. If **-reference_rule_name** is not specified, then the driver taper over pin layers defaults to unspecified, and the tool copies the value from **-taper_over_pin_layers**.

-driver_taper_under_pin_layers num_of_layers ***num_of_layers***

Specifies the number of routing layers under the output pin layer that the router may use for tapering, and enables layer-based pin tapering.

Values must be greater or equal to 0. If both the **-driver_taper_over_pin_layers** and **-driver_taper_under_pin_layers** options are set to zero, pin tapering is disabled on the pin layer. If either option is set to a nonzero value, the pin layer is considered a taper layer.

If this option is not specified for this rule, the tool copies it from the reference rule specified by **-reference_rule_name**. If **-reference_rule_name** is not specified, then the driver taper under pin layers defaults to unspecified, and the tool copies the value from **-taper_over_under_layers**.

-spacings {*layer_name* **{***spacing* **...} ...}**

Specifies the minimum different-net spacing allowed between wires for each named routing layer. Each entry in the list is comprised of a routing layer name and its spacing separated by a space. The spacing portion of the pair is a list of floating point numbers. If you specify more than one spacing, you must specify the **-spacing_weight_levels** option. The spacing values in the **-spacings** argument and the weight values in the **-spacing_weight_levels** argument must have a one-to-one correspondence.

If you do not specify the minimum spacing for a layer, the tool copies the minimum spacing from the reference rule specified by **-reference_rule_name**. If **-reference_rule_name** is not specified, then the minimum spacing defaults to unspecified for the layer.

-spacing_weight_levels {*layer_name* **{***low***|***medium***|***high***|***hard* **...} ...}**

Sets the weight for the corresponding spacing that is specified in the **-spacings** option. Each entry in the list is comprised of a routing layer name and its weight level separated by a space. Each weight value must be either of low, medium, high or hard. Following values are used for conversion: low = 0.3, medium = 0.5, high = 0.7, hard = 1. There can be maximum one hard spacing rule per routing layer.

-spacing_length_thresholds {*layer_name* **{***spacing_length_threshold* **...} ...}**

Sets the length threshold for the corresponding spacing that is specified in the **-spacings** option. Each entry in the list is comprised of a routing layer name and its length threshold list in microns, separated by a space. The length threshold portion of the pair is a list of floating point numbers.

The length threshold values in the **-spacing_length_thresholds** argument and the spacing values in the **-spacings** argument must have a one-to-one correspondence.

Router reports and tries to fix only those DRC violations where the parallel length of the metal involved in the violation over- laps by at least the specified threshold. Router calculates the parallel length using a shape-based formulation. The purpose of this option is to increase the flexibility of DRC convergence but not hurt crosstalk in a significant way.

The default threshold value is 0.

-multiplier_width ***width_multiplier***

Specifies the multiplier to apply to the layer widths in this routing rule. For layer widths specified by the **-widths** option, the widths are multiplied by width_multiplier. For each layer not specified in the **-widths** option, the default routing rule's layer width (i.e., the layer's default width in the tech file) is multiplied by width_multiplier and set as the layer width in this routing rule. You do not

need to know the units or layer width currently defined in the default routing rule. For example, to define double width, specify **-multiplier_width 2.0**.

-multiplier_spacing ***spacing_multiplier***

Specifies the multiplier to apply to the layer spacings in this routing rule. For layer spacings specified by the **-spacings** option, the spacings are multiplied by spacing_multiplier. For each layer not specified in the **-spacings** option, the default routing rule's layer spacing (i.e., the layer's minimum spacing in the tech file) is multiplied by spacing_multiplier and set as the layer spacing in this routing rule. You do not need to know the units or layer spacing currently defined in the default route rule. For example, to define double spacing, specify **-multiplier_spacing 2.0**.

Each multiplied spacing is assigned a hard weight level. The spacing_multiplier is only applied to hard rules. So if a layer is specified in the spacing_weight_levels option with a non-hard weight, the spacing_multiplier is not applied to any of that particular layer's spacings, because the spacing_multiplier cannot be applied to soft rules.

-single_side_spacing

Applies the spacing rule to only one side of the net, instead of both sides of the net.

-ignore_spacing_to_pg true|false

Indicates whether or not the router should ignore this routing rule's spacing constraints when considering the distance to power and ground nets. If not specified, the router uses the setting specified by the **route.common.ignore_var_spacing_to_pg** application option.

-ignore_spacing_to_blockage true|false

Indicates whether or not the router should ignore this routing rule's spacing constraints when considering the distance to blockages. If not specified, the router uses the setting specified by the **route.common.ignore_var_spacing_to_blockage** application option.

-ignore_spacing_to_shield true|false

Indicates whether or not the router should ignore this routing rule's spacing constraints when considering the distance to shield nets. If not specified, the router uses the setting specified by the **route.common.ignore_var_spacing_to_shield** application option.

-single_connection_to_pin true|false

Indicates whether or not the router must connect to a pin only once. If true, the router must connect to a pin only once. This applies to all pins connected to nets with this routing rule. If false, the router is allowed to connect to a pin any number of times. If this option is unspecified, the default is false except for nets using a nondefault width routing rule. For those nets, the value is always true. This option takes precedence over the global app option "route.common.single_connection_to_pins".

-rdl_taper_distances {*layer_name distance* **...}**

Specifies the maximum length in microns for tapering wires in the redistribution layer. The default is 0 and the wires are not tapered. This option is supported only by the flip-chip router.

-rdl_taper_widths {*layer_name width* **...}**

Specifies the tapering width for each redistribution layer. If you do not specify this option, the flip-chip router uses the size of the via or pin as the wire width. This option is supported only by the flip-chip router.

-mask_constraints {{*layer_name mask_name* **fixed} ...}**

Specifies the mask constraint for each routing layer. Each entry in the option value consists of the routing layer name, its mask constraint, and the optional **fixed** keyword. The **fixed** keyword is specified if and only if the mask constraint is fixed on the layer (i.e., it is a hard constraint). If the "fixed" keyword is not specified, the mask constraint is not fixed on the layer (i.e., it is a soft constraint). The supported masks are mask_one, mask_two, and same_mask.

If you do not specify the mask constraint for a routing layer, the tool copies the mask constraint from the reference rule specified by **-reference_rule_name**. If **-reference_rule_name** is not specified, there is no default value.

-via_spacings {{*layer_name layer_name spacing*} ...}

Specifies the via spacing between two via layers. The two via layers can be the same layer or different layers. Only one via spacing may be specified for each pair of via layers.

The minimum spacing between vias on any layer combinations not specified in this option is determined from the technology file. For vias on the same layer, the router uses the minimum spacing defined in the Layer section for the via layer. For vias on different layers, the router uses the minimum spacing defined in the DesignRule section for the via layer combination.

-parallel_wire {*layer_name* ...}

Specifies a list of layers on which to create fat wires for parallel wire purpose. Layers must be interconnect or via_cut layers.

rule_name

Specifies the name for the new non-default routing rule. This is a required argument.

DESCRIPTION

This command defines non-default routing rules in the design.

Use the **create_routing_rule** command to define width, shielding, tapering, and spacing rules.

Use the **set_routing_rule** command to set the minimum and maximum routing layers, and to assign non-default routing rules to specific nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines a non-default routing rule named new_rule that uses the default routing rules as the reference rule:

```
prompt> create_routing_rule new_rule -default_reference_rule \
  -widths {m1 0.8 m4 0.9} -spacings {m1 1.0 m4 1.0}
```

The following example reports the non-default rule named new_rule:

```
prompt> create_routing_rule new_rule
prompt> report_routing_rules new_rule
```

The following example defines a non-default rule named new_rule that specifies spacings, spacing weights, and spacing length thresholds:

```
prompt> create_routing_rule new_rule \
  -spacings {m1 {0.09 0.15 0.2} m2 {0.09 0.15 0.2} m3 {0.09 0.15 0.2}} \
  -spacing_weight_levels {m1 {high medium low} m2 {high medium low} \
            m3 {high medium low}} \
  -spacing_length_thresholds {m1 {0.01 0 0} m2 {0.01 0 0} m3 {0.01 0 0}}
```

The following example defines a non-default rule named new_rule that specifies multiple via pattern with both rotated and unrotated vias for certain layers.

```
prompt> create_routing_rule new_rule \
  -vias {{VIA1_HH 2x1 NR} {VIA1_HH 2x1 R} {VIA1_HV 2x1 R} {VIA1_HV 2x1 R}}
```

The following example defines a non-default rule named new_rule that specifies layer-based tapering options.

```
prompt> create_routing_rule new_rule -taper_over_pin_layers 1 -taper_under_pin_layers 3
```

SEE ALSO

remove_routing_rules(2)
report_routing_rules(2)
set_routing_rule(2)
route.common.ignore_var_spacing_to_blockage(3)
route.common.ignore_var_spacing_to_pg(3)
route.common.ignore_var_spacing_to_shield(3)

create_rp_group

Creates a relative placement group in the given design.

SYNTAX

collection **create_rp_group**
-name *rp_group_name*
[-rows number_of_rows]
[-columns number_of_columns]
[-design *design*]

Data Types

rp_group_name string
number_of_rows integer
number_of_columns integer
design collection

ARGUMENTS

-name *rp_group_name*

Specifies the name of the relative placement group.

-rows *number_of_rows*

Specifies the number of rows into which objects can subsequently be added, relative to each other. The default value for the option is 1.

-columns *number_of_columns*

Specifies the number of columns into which objects can subsequently be added, relative to each other. The default value for the option is 1.

-design *design*

Specifies the top design for creating relative placement group. If this is not specified then relative placement group will be created in the current design.

DESCRIPTION

The **create_rp_group** command creates new relative placement group. This command returns a collection containing the created relative placement group. If no objects are created, the empty string is returned.

The created group is empty and contains no objects. The objects to relative placement group can be added using the **add_to_rp_group** command.

Relative placement groups provide for explicit user control of placement for a group of cells. A relative placement group is placed as a whole while maintaining the relative placement (or tiling) of the cells within the group as specified with the **add_to_rp_group** command. Relative placement is also known as "structured placement."

When placing a relative placement group, the placer automatically chooses the orientation for the group. For example, if the placer chooses the north orientation for a relative placement group, the first column of that relative placement group is placed first, and subsequent columns are placed toward the right. If the placer chooses the flip-north orientation then last column will be placed first and other columns towards the right.

EXAMPLES

The following example creates a relative placement group named "top_rp", with 3 rows and 4 columns.

```
prompt> create_rp_group -name top_rp -rows 3 -columns 4
```

SEE ALSO

create_rp_group(2)
get_rp_groups(2)
remove_rp_groups(2)
add_to_rp_group(2)
remove_from_rp_group(2)
set_rp_group_options(2)
write_rp_groups(2)

create_sadp_track_rule

Creates a self-aligned double patterning (SADP) track rule.

SYNTAX

status **create_sadp_track_rule**
-name *track_pattern_rule*
-pattern *pattern*
-sadp_spacing *spacing*

Data Types

track_pattern_rule string
pattern list
spacing float

ARGUMENTS

-name *track_pattern_rule*

Defines the name of the track rule. As multiple *track_pattern_rules* can be created for the track plan, the rule name should be unique in the current library.

-pattern *pattern*

Defines the inter-leaving wide tracks which can be used either for specific PG nets or left without any net assignments. If no nets are assigned, those wide tracks can be used later by nets with nondefault routing rules for wide metal routing. The following are example pattern specifications:

{{width1 count1 netname1} {{width2 count2 netname2} ... }
{{width1 count1 netname1} {{width2 count2 } ... }

The fields used in the pattern specification are:

- width - The width of the track used by netname.
- count - The number of default width tracks allowed until the next wide track defined by width.
- netname - Net which can be routed over this track. This field is optional. If specified unrouted_net with name netname should exist in the block where sadp_tracks will be created.

-sadp_spacing *spacing*

Defines the minimum spacing between the adjacent tracks created by this track rule. This spacing along with default or minimum width of the route is used to calculate the track position of the default width tracks between width1, width2, and so on.

DESCRIPTION

This command defines a self-aligned double patterning (SADP) track rule by specifying SADP constraints. The SADP rules are used to create SADP tracks. These rules are persistent across sessions and are saved in the current library.

EXAMPLES

The following example creates a self-aligned double patterning (SADP) track rule named tr1. For net VDD, the track width is 1 and the track count is 1. For net VSS, the track width is 2 and the track count is 1. The minimum spacing between adjacent tracks is 0.50.

```
prompt> create_sadp_track_rule -name tr1 \
  -pattern {{1 1 "VDD"} {2 1 "VSS"}} -sadp_spacing 0.50
```

The following example creates a self-aligned double patterning (SADP) track rule named tr2 for net VSS and other unassigned nets. For VSS, the track width is 2 and the track count is 1. For other nets, the track width is 2 and the track count is 1. The minimum spacing between adjacent tracks is 0.50.

```
prompt> create_sadp_track_rule -name tr2 -pattern {{2 1 "VSS"} {2 1}} \
  -sadp_spacing 0.50
```

SEE ALSO

check_sadp_tracks(2)
generate_sadp_tracks(2)
remove_sadp_track_rule(2)
report_sadp_track_rule(2)

create_safety_register_group

Creates one safety_register_group object

SYNTAX

status **create_safety_register_group**
-rule *safety_register_rule*
[-name *group_name*]
[-registers *objects*]
[-logic *cells*]
[-taps *cells*]
[-split_pins *pins*]

Data Types

safety_register_rule object
group_name string
objects collection
cells collection
pins collection

ARGUMENTS

-rule *safety_register_rule*

Specifies the safety registery rule based on which this redundancy group was created

-name *group_name*

Name of the new safety_register_group object. If not specified, the name will be automatically generated by the tool

-registers *objects*

Specifies the collection of registers or register output pins which should be part of this register redundancy group.

-logic *cells*

Specifies the collection of logic cells which form a part of the voting logic circuit. These must be combinational cells.

-taps *cells*

Specifies the collection of tap cells for the registers in this group.

-split_pins *pins*

Specifies the set of pins which should be split so as to be a part of different branches of high-fanout tree.

DESCRIPTION

This command creates one safety_register_group object based on the given parameters. It stores the reference to the original safety register rule which was honored to create it.

EXAMPLES

The following example creates a safety_register_group object consisting of 3 registers and some voting logic combinational cells.

```
prompt> create_safety_register_group -rule rule1 \
   -registers {flop1 flop2 flop3} \
   -logic {land1 land2 lxor1} \
   -name group1
```

The following example creates a safety_register_group object consisting of 3 register output pins and tap cells

```
prompt> create_safety_register_group -rule rule2 \
   -registers {flop1/Q flop2/Q flop3/Q} \
   -taps {TAP1 TAP2} \
   -name group2
```

SEE ALSO

get_safety_register_groups(2)
remove_safety_register_groups(2)
report_safety_register_groups(2)
write_safety_register_script(2)

create_safety_register_rule

Creates one safety_register_rule object

SYNTAX

status **create_safety_register_rule**
-type *rule_type*
[-name *rule_name*]
[-distance *values*]
[-register_mapping *lib_cell_list*]
[-logic_module *reference*]
[-tap_mapping *lib_cell_list*]
[-split_pin_types *pin_type_list*]

Data Types

rule_type	string
reference	object
values	list
lib_cell_list	list
pin_type_list	list

ARGUMENTS

-type *rule_type*

Specifies whether the register should be mapped to another fault-tolerant variant, or should be replaced by a group of redundant flops (redundancy group). Valid values are "fault_tolerant", "dual_mode" and "triple_mode"

In case of redundancy group, it specifies the module that the register should be replaced by.

-name *rule_name*

Optionally specifies the name of the rule to be created. If not specified, a name will be automatically generated and assigned.

-distance *values*

Specifies the distance values for repelling bound constraint for redundancy group. It could be either in the form of an {x y} pair or a single number denoting radial distance. The values are in user units. The values cannot be negative. The distances are measured edge-to-edge, or in other words, from the boundary of one register to the closest boundary of another. For *-type dual_mode or triple_mode*, this option must be specified. For *-type fault_tolerant*, this option is unnecessary and will be ignored.

Specifying the {x y} distance pair means that either "x" or "y" separation must be honored. In other words, each pair of registers in the group should be either "x" distance apart measured on X-axis, or "y" distance apart measured on Y-axis. This will effectively create a rectangular keep-out region around each register, in which it is forbidden to place another register of the same group. It

is allowed to specify a distance of zero for "x" or "y", but not for both simultaneously.

Specifying the single "r" value means that the Manhattan distance between each pair of registers in the group should be at least "r". The Manhattan distance between two redundancy registers is defined as the sum of the vertical distance and horizontal distance between their closest edges. Specifying a radial distance thus creates an octagonal keep-out region around each register, wherein another register of the same group must not be placed. For cells in the same column, the horizontal distance is 0. For cells in the same row, the vertical distance is 0. This "r" value cannot be zero.

-register_mapping *lib_cell_list*

Specifies one or more lib_cells that should be considered for safety registers of type fault tolerant. When this option is provided, synthesis will map the original register to an appropriate lib_cell from the given values.

This option cannot be specified with *-type dual_mode* or *triple_mode*. This option cannot be specified along with *-logic_module*.

-logic_module *reference*

Specifies the module containing the combinational logic which follows the duplicated or triplicated registers after the safety register group is created based on the given rule. For example, in case of triple-mode redundancy, this could be a majority circuit with three inputs and one output.

This option cannot be specified with *-type fault_tolerant*. This option cannot be specified along with *-register_mapping*.

-tap_mapping *lib_cell_list*

The list of lib cells, out of which the appropriate one will be used for instantiating tap cells for the safety registers.

-split_pin_types *pin_type_list*

The pin types which should be considered for splitting so as to be a part of different branches of clock tree or high fanout net trees. Allowed values are *clock*, *reset* and *scan*. List of one or more such types can be specified.

DESCRIPTION

This command creates a new safety_register_rule object based on the options specified. This rule can be then applied to safety-critical registers. The rule can specify replacement of the register either to another fault-tolerant register, or by a group of redundant registers. In latter case, the minimum separations between the redundant registers can be specified which can be considered by the placement engine. If the pins of the redundant registers should be split to different branches of trees, the types to be considered can also be specified.

EXAMPLES

The following example creates a safety_register_rule object.

```
prompt> create_safety_register_rule -type fault_tolerant \
   -distance {20 30} \
   -split_pin_types {scan clock} \
   -name rule1
```

The following example defines a rule that requires redundancy registers to be separated by at least 20u horizontally, or, at least 5u vertically.

```
prompt> create_safety_register_rule -type triple_mode
```

-name RULE1 -distance {20 5}

Assuming that the width of current block is smaller than 2000u, following rule states that redundancy registers must be placed into separate rows. Because the specified y distance is zero, adjacent rows are allowed.

```
prompt> create_safety_register_rule -type triple_mode\
  -name RULE2 -distance {2000 0}
```

Assuming that the width of current block is smaller than 2000u, following rule states that redundancy registers must be placed into separate rows. The vertical between the two redundancy registers must be at least 10u.

```
prompt> create_safety_register_rule -type dual_mode\
  -name RULE3 -distance {2000 10}
```

The following rule defines a required Manhattan distance of 8u between the redundancy registers that should follow this rule.

```
prompt> create_safety_register_rule -type triple_mode -name RULE4 -distance 8
```

Note: All examples above assume the length unit to be microns.

SEE ALSO

create_safety_register_group(2)
get_safety_register_rules(2)
remove_safety_register_rules(2)
report_safety_register_rules(2)
set_safety_register_rule(2)
write_safety_register_script(2)

create_scan_chain

Creates a scan chain in the current design.

SYNTAX

collection **create_scan_chain**
-name *name*
[-availabe_bits *number_of_bits*]
[*stub_chains*]

Data Types

name string
number_of_bits integer
stub_chains collection

ARGUMENTS

-name *name*

Name of the scan_chain

-partition *name*

Scan partition name

-available_bits *number_of_bits*

Number of bits available with respect to the expected bit length of this scan_chain

stub_chains

Ordered collection of stub_chain objects to be added into the new scan_chain

DESCRIPTION

This command creates a new scan chain and returns a collection containing the new scan chain.

EXAMPLES

The following example creates a scan chain with name scan1 and available_bits 8

```
prompt> create_scan_chain -name scan1 -available_bits
```

The following example creates a scan chain with name scan2 and stub_chains {stub1 stub2}

```
prompt> create_scan_chain -name scan2 {stub1 stub2}
```

SEE ALSO

get_scan_chains(2)
remove_scan_chains(2)

create_scenario

Creates a scenario in the current design.

SYNTAX

collection **create_scenario**
-mode *mode*
-corner *corner*
[-name *name*]
[-init_from *corner_or_scenario*]

Data Types

mode	collection
corner	collection
name	string
corner_or_scenario	collection

ARGUMENTS

-mode *mode*

Specifies the mode of the new scenario. This option is required. The mode must already exist.

-corner *corner*

Specifies the corner of the new scenario. This option is required. The corner must already exist.

-name *name*

Specifies the name of the new scenario. If this option is not given, a unique name will be synthesized from the mode and corner names.

-init_from *corner_or_scenario*

Specifies an existing scenario or corner object. If given, this object will be used to initialize the new scenario's constraints and analysis flags. If a scenario is given, it must have the same mode as the scenario being created. If a corner is given, there must be an existing scenario with that corner and the mode of the scenario being created.

DESCRIPTION

Creates a scenario in the current design, and sets it active for all analysis types except cell and signal EM. If any of these analysis

types are not needed, this command should be followed by the **set_scenario_status** command. If a scenario already exists for the given mode and corner, the command will fail. If a scenario with the given name already exists, the command will fail. If a scenario is successfully created, its mode will become the current mode, and its corner will become the current corner. This means that the new scenario will be the current scenario.

The **-init_from** option allows you to duplicate all the constraints and settings of an existing scenario onto a new scenario. If an existing scenario is given with **-init_from**, it will be used as the source. If a corner is given, it will be used (along with the mode given by the **-mode** option) to search for an existing scenario to serve as the source. In either case, the source scenario must have the same mode as the new scenario.

If successful, the command returns a collection containing the new scenario.

Multicorner-Multimode Support

By default, this command uses the current mode and current corner. To specify a different mode, use the **-mode** option. To specify a different corner, use the **-corner** option.

EXAMPLES

The following command creates a scenario named m1@c1 using corner c1 and mode m2.

```
prompt> create_scenario -name m1@c1 -mode m1 -corner c1
```

SEE ALSO

create_corner(2)
create_mode(2)
remove_scenarios(2)
report_scenarios(2)
set_scenario_status(2)

create_secondary_pg_placement_constraints

Create secondary pg placement constraints that will allow dual rail cells to be inserted into specific regions.

SYNTAX

status **create_secondary_pg_placement_constraints**
[-name *constraint_name*]
[-supply *supply_net_name*]
[-exclude_supply *exclude_supply_net_name*]
[-voltage_areas *voltage_area_list*]
[-layers *layer_list*]
[-margin *distance*]
[-region *region_list*]

Data Types

constraint_name string
supply_net_name string
exclude_supply_net_name string
voltage_area_list collection
layer_list list of strings
distance distance in user units
region_list list of coord

ARGUMENTS

-name

Specifies the name of this pg constraint. The name of the constraint should be unique for the design. No two secondary pg constraints should have the same name.

-supply

This is optional and specifies the supply net name. Either this option or -exclude_supply should be specified, but not both.

-exlude_supply

This is optional and specifies the excluded supply net name. Either this option or -supply should be specified, but not both.

-voltage_areas

Specifies the list of voltage areas (VA) that this pg constraint will be applied to. When -voltage_areas is not specified, this pg constraint will be applied to all voltage areas in the current block, even if the same supply has restricted physical availability in another VA. PG straps in a foreign VA that is not included in the list of VAs in the constraint for the same supply will not be used for buffering, even if the strap in the foreign VA is available within the specified margin of the VA listed in the constraint.

-layers

Specifies the list of layers. The physical shapes where the dual rail cells should be inserted into are derived based on the pg straps on the specified layers with shape_use of "stripe". This optional should not be specified with -exclude_supply, or an error will be issued.

-margin

This is optional and specifies the margin distance when -layers is used, which is a floating point number specifying the maximum distance between the outer edge of a strap and the furthest edge of a dual-rail cell in the distance unit of the design. The cells will be placed such that no part of it will be outside the margin. We recommend the margin be large enough such that when each edge of each strap is expanded by the margin amount, neighboring expanded straps should touch or overlap, resulting in a continuous region for dual rail cell placement. The default margin is 0. When margin is specified and has a none zero value, -layers must be specified, or an error will be issued. This optional should not be specified with -exclude_supply, or an error will be issued.

-region

Specifies the rectanges or rectilinear polygons where the dual rail cells should be inserted. This use model can be used even if power network synthesis has not been completed. If -margin or -layers or -voltage_area is specified with -region, a warning will be issued and the -margin/-layers/-voltage_areas options will all be ignored. This optional should not be specified with -exclude_supply, or an error will be issued.

Each region can be one of the following objects:

- Rectangle
  Each rectangle is specified by its lower-left and upper-right coordinates as {*llx lly*} {*urx ury*}).
- Polygon
  Each polygon is specified by its points as {*x1 y1*} {*x2 y2*} {*x3 y3*} {*x4 y4*} and so on.
- Geometric object A geometric object is the combined area of a mixed collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area include the areas of each object. In the case of layers, the resulting area includes the area of every shape on the layer. Overlapping areas are preserved and not merged.

The specified regions can overlap or be disjoint.

DESCRIPTION

This command creates a secondary pg constraint. When there is a secondary pg constraint with -supply being defined for certain supplies, this implies that these supplies will not be available everywhere in the voltage area/power domain as what UPF specified. Instead, they will only be available within specific regions which the secondary pg constraint specified. When there is a secondary pg constraint with -exclude_supply being defined for certain supplies, this implies that these supplies will not be available for implementation in the voltage areas specified even though these supplies are indicated as available in UPF.

When a secondary pg constraint with -supply is creatd, the requirement to place the dual-rail cells in the regions with the power straps is a hard constraint. If regions with straps are not available where needed, nets can remain unbuffered (logical design rule check issues such as transition violations, or timing violations). Dual-rail cells including buffers, inverters, isolation and level shifters are considered to honor the constraint. Power switch cells are not considered. This constraint is for cases where PG straps are available in some region of a VA.

When a secondary pg constraint with -exclude_supply is created, the tool should not insert any cells that require the excluded supply in the specified voltage area/power domains. This is also a hard constraints. This constraint is for cases where supplies are available in UPF, but there are no PG straps planned and implemented in the corresponding voltage areas.

This feature supports the flat flow with VAs in the current_design. Hierarchical flows with restricted secondary PG straps in VAs that are in linked physical blocks are not supported. Only the advanced legalizer is supported with this feature. It must be enabled to use this feature. For optimization, only the default "advanced buffering" is supported.

This feature uses VA shapes to implement the placement constraints under the hood. VA shapes will be generated based on the secondary PG strap location so that dual-rail cells can be placed in the strap region. The VA shapes created by the user can be modified in number and size. There are attributes to distinguish VA shapes created as a result of secondary PG placement constraints from regular VA shapes. Each generated VA shape represents a region where straps for one or more supplies are available. Do not alter or remove the VA shapes related to the secondary PG placement constraints.

Exactly one of either -layers or -region must be specified if -supply is specified. If multiple calls of create_secondary_pg_placement_constraints for the same supply and voltage area are issued, the union of the locations is considered.

The constraint is saved with the design in the nlib.

EXAMPLES

The following example creates a secondary pg constraint named pg_cstr0 that defines supply VDD2 being available only within a margin of 1 from the VDD2 straps in voltage area VA on layer M7 and M8 after commit_secondary_pg_placement_constraints is issued.

```
prompt> create_secondary_pg_placement_constraints -name pg_cstr0 \
  -supply VDD2 -voltage_areas VA -margin 1 -layers {M7 M8}
prompt> commit_secondary_pg_placement_constraints
```

The following example creates a secondary pg constraint named pg_cstr1 that defines supply VDD1 being available only within region {{120 30 } {250 70}} after commit_secondary_pg_placement_constraints is issued.

```
prompt> create_secondary_pg_placement_constraints \
  -name pg_cstr1 -supply VDD1 -region {{120 30 } {250 70}}
prompt> commit_secondary_pg_placement_constraints
```

The following example creates a secondary pg constraint named pg_cstr2 that defines supply VDD1 to be excluded from implementation usage in all voltage areas after commit_secondary_pg_placement_constraints is issued.

```
prompt> create_secondary_pg_placement_constraints \
  -name pg_cstr2 -_exclude_supply VDD1
prompt> commit_secondary_pg_placement_constraints
```

SEE ALSO

commit_secondary_pg_placement_constraints(2)
remove_secondary_pg_placement_constraints(2)
report_secondary_pg_placement_constraints(2)
check_bufferability(2)
place.legalize.enable_advanced_legalizer(3)
opt.buffering.enable_advanced_buffering(3)
voltage_area_shape_attributes(3)

create_shape

Creates a shape object in the current design.

SYNTAX

collection **create_shape**
-shape_type rect | polygon | path | text
-layer *layer*
[-shape_use *use*]
[-net *net* | -port *port*]
[-boundary { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects]
[-path {{*x y*} {*x y*}} ...]
[-width *width*]
[-start_endcap *endcap_type*]
[-end_endcap *endcap_type*]
[-start_extension *extension*]
[-end_extension *extension*]
[-origin *point*]
[-height *height*]
[-orientation *orientation*]
[-justification *justification*]
[-text *text_string*]
[-fill_cell *fill_cell*]

Data Types

layer collection
use string
llx float
lly float
urx float
ury float
x float
y float
geometric_objects collection
width float
endcap string
extension float
height float
orientation string
justification string
text_string string
net collection
port collection
fill_cell collection

ARGUMENTS

-shape_type rect | **polygon** | **path** | **text**

Specifies the type of shape to create.

This is a required option.

-layer ***layer***

Specifies the layer on which to create the shape. You must specify a single layer using either the layer name from the technology file or a collection containing exactly one layer.

This is a required option.

-shape_use ***use***

Specifies the usage of the shape.

Valid values are

- **area_fill**
  A fill shape that does not require optical proximity correction (OPC). Fill shapes are created inside the top level **fill_cell** for the design, unless a **fill_cell** is specified using **-fill_cell** option.
- **core_wire**
  A shape that connects the endpoints of a follow-pin to its target.
- **detail_route** (the default)
  A shape used for detail routed signal nets.
- **follow_pin**
  A shape that connects a standard cell to a power structure.
- **global_route**
  A shape used for global routed signal nets.
- **lib_cell_pin_connect**
  A shape that connects to a pin of a standard cell.
- **macro_pin_connect**
  A shape that connects to a pin of an I/O pad cell or macro cell.
- **opc**
  A fill shape that requires optical proximity correction (OPC).
- **pg_augmentation**
  A shape added post route to the power and ground structures.
- **ring**
  A shape that belongs to a power ring.
- **shield_route**
  A shape used for shielding route shapes.
- **stripe**
  A shape that belongs to a power strap.

- **user_route**
  A shape used for user-entered route shapes.
- **zero_skew**
  A shape used for zero-skew route shapes.

This option is valid only for rectangle, polygon, and path shapes; the tool issues an error if you use it for text shapes.

-net ***net***

Specifies the net that owns this shape. If the net is power or ground, the shape rule_type attribute is automatically set to "none". Otherwise it is set to "default".

This option is valid only for rectangle, polygon, and path shapes; the tool issues an error if you use it for text shapes.

The **-net** and **-port** options are mutually exclusive; you can specify neither option (for unassigned shapes) or one of them.

-port ***port***

Specifies the port that owns this shape.

When you assign a shape to a port, the tool creates a new terminal named TM_*n*, where *n* is a unique integer in the block. The new terminal is owned by the port, owns the shape, and has the default access direction (all).

This option is valid only for rectangle, polygon, and path shapes; the tool issues an error if you use it for text shapes.

The **-net** and **-port** options are mutually exclusive; you can specify neither option (for unassigned shapes) or one of them.

-boundary ***list_of_points***

Specifies the boundary of a rectangle, polygon or path shape. The boundary may be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates (i.e., {*llx lly*} {*urx ury*}). A polygon is specified by its points (i.e., {*x y*} {*x y*} {*x y*} {*x y*}...).

Polygons may also be specified as the combined area of a heterogeneous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

When specifying the boundary polygon as a collection of physical objects, the resulting area must resolve to a single, connected polygon. It is an error to specify a collection of objects with a combined area that resolves to multiple polygons.

This option is required for rectangle and polygon shapes. It is an error to use this option with other shape types.

-path {{*x y*} {*x y*} ...}

Specifies the centerline points of a path shape. You must specify at least two points.

If you use this option with other shape types, the tool issues an error.

-width ***width***

Specifies the width of a path shape. The value must be non-negative.

If you use this option with other shape types, the tool issues an error.

-start_endcap ***endcap_type***

Specifies the type of the start endcap of a path shape.

The valid values are

- **flush** (the default)

Creates an endcap with square ends and no extension.

- **full_width**
  Creates an endcap with square ends that is extended by the path width.
- **half_width**
  Creates an endcap with square ends that is extended by half the path width.
- **octagon**
  Creates an endcap with octagonal ends and no extension.
- **variable**
  Creates an endcap with square ends that is extended by the value specified in the **-start_extension** option. When you specify this setting, the **-start_extension** option is required.

This option is valid only for path shapes; the tool issues an error if you use it for other shape types.

-end_endcap *endcap*

Specifies the type of the end endcap of a path shape.

The valid values are

- **flush** (the default)
  Creates an endcap with square ends and no extension.
- **full_width**
  Creates an endcap with square ends that is extended by the path width.
- **half_width**
  Creates an endcap with square ends that is extended by half the path width.
- **octagon**
  Creates an endcap with octagonal ends and no extension.
- **variable**
  Creates an endcap with square ends that is extended by the value specified in the **-start_extension** option. When you specify this setting, the **-start_extension** option is required.

This option is valid only for path shapes; the tool issues an error if you use it for other shape types.

-start_extension *extension*

Specifies the start extension of a path shape with variable endcaps. The value must be non-negative.

This option is required when you use the **-start_endcap variable** option. If you use this option with other shape types, the tool issues an error.

-end_extension *extension*

Specifies the end extension of a path shape with variable endcaps. The value must be non-negative.

This option is required when you use the **-end_endcap variable** option. If you use this option with other shape types, the tool issues an error.

-origin {*x y*}

Specifies the placement location of the text shape's origin. The origin is determined by the text's justification, as specified with the **-justification** option.

This option is required for text shapes. If you use this option with other shape types, the tool issues an error.

-height *height*

Specifies the height of a text shape. The value must be non-negative.

This option is required for text shapes. If you use this option with other shape types, the tool issues an error.

-orientation *orientation*

Specifies the orientation of a text shape.

Valid values are

- **MX** (mirror around the X-axis)
- **MXR90** (mirror around the X-axis and rotate 90 degrees)
- **MY** (mirror around the Y-axis)
- **MYR90** (mirror around the Y-axis and rotate 90 degrees)
- **R0** (no rotation, which is the default)
- **R90** (rotate 90 degrees)
- **R180** (rotate 180 degrees)
- **R270** (rotate 270 degrees)

This option is valid only for text shapes; the tool issues an error if you use it for other shape types.

-justification *justification*

Specifies the justification of a text shape. Text is justified with respect to the untransformed bounding box (the bounding box before applying the orientation rotation).

Valid values are

- **C** (center)
- **CB** (center-bottom)
- **CT** (center-top)
- **LB** (left-bottom, which is the default)
- **LC** (left-center)
- **LT** (left-top)
- **RB** (right-bottom)
- **RC** (right-center)
- **RT** (right-top)

This option is valid only for text shapes; the tool issues an error if you use it for other shape types.

-text *text_string*

Specifies the text string to display in a text shape.

This option is required for text shapes. If you use this option with other shape types, the tool issues an error.

-fill_cell *fill_cell*

Specifies the *fill_cell* to create the shape in. The shapes are created as fill shapes even if **-shape_use** is not specified as **area_fill**

DESCRIPTION

This command creates a new rectangle, polygon, path, or text shape and returns a collection that contains the new shape.

For rectangle (**-shape_type rect**) and polygon (**-shape_type polygon**) shapes, you must specify the layer (**-layer**) and boundary (**-boundary**). You can also specify the use (**-shape_use**) and ownership (**-net** or **-port**) for the shape.

For path (**-shape_type path**) shapes, you must specify the layer (**-layer**) with either combination of path (**-path**) and width (**-width**) or boundary (**-boundary**) with points that form rectangle. You can also specify the use (**-shape_use**), ownership (**-net** or **-port**), start endcap (**-start_endcap**), and end endcap (**-end_endcap**) for the shape. If you specify the **-start_endcap variable** option, you must also specify the **-start_extension** option. If you specify the **-end_endcap variable** option, you must also specify the **-end_extension** option.

For text (**-shape_type text**) shapes, you must specify the layer (**-layer**), origin (**-origin**), height (**-height**), and text string (**-text**). You can also specify the orientation (**-orientation**) and justification (**-justification**) for the shape.

EXAMPLES

The following example creates a rectangle shape on the port named Clk.

```
prompt> create_shape -shape_type rect -layer METAL \
  -boundary {{0 0} {1000 1000}} -port Clk
```

The following example creates a polygon shape that is not associated with any net or port.

```
prompt> create_shape -shape_type polygon -layer METAL \
  -boundary {{0 0} {0 1000} {500 1000} {500 0}}
```

The following example creates a path shape on the net named Clk.

```
prompt> create_shape -shape_type path -layer METAL \
  -path {{0 0} {0 1000}} -width 50 -net Clk
```

The following example creates a path shape.

```
prompt> create_shape -shape_type path -layer METAL \
  -boundary {{0 0} {0 1000}}
```

The following example creates a text shape that contains the string "hello world".

```
prompt> create_shape -shape_type text -layer METAL \
  -origin {100 100} -height 100 -orientation R0 -justification LB \
  -text "hello world"
```

SEE ALSO

get_shapes(2)

remove_shapes(2)

create_shape_pattern

Creates a shape_pattern object in the current design.

SYNTAX

collection **create_shape_pattern**
-shape_pattern_type rect_pattern | polygon_pattern | path_pattern
-layer *layer*
-net *net*
[-shape_use *use*]
[-base_boundary { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects]
[-base_path {{*x y*} {*x y*}} ...]
[-base_width *width*]
[-base_start_endcap *endcap*]
[-base_end_endcap *endcap*]
[-base_start_extension *extension*]
[-base_end_extension *extension*]
[-m_dimension *dimension*]
[-n_dimension *dimension*]
[-m_displacement *delta*]
[-n_displacement *delta*]
[-displacements *delta_list*]
[-grid_size *gridSize*]
[-occupancy_pattern *occupancy_pattern*]
[-mask_pattern *mask_pattern*]
[-uniform_mask_constraint *uniform_mask*]
[-alternating_mask_constraints *mask_list*]
[-is_uniform_mask_fixed *true* | *false*]

Data Types

layer collection
use string
geometry { {llx lly} {urx ury} }
| { {x y} {x y} {x y} {x y} ... }
| geometric_objects
llx float
lly float
urx float
ury float
x float
y float
geometric_objects collection
width float
endcap string

extension float
net collection
dimension positive-integer
delta distance:direction | {x-distance:x-direction
y-distance:y-direction}
distance float
direction string
delta_list { delta1 delta2 ... }
gridSize positive-integer
occupancy_pattern string
mask_pattern string
uniform_mask { mask [fixed] }
mask_list { mask1 mask2 ... }

ARGUMENTS

-shape_pattern_type rect_pattern | polygon_pattern | path_pattern

Specifies the type of shape_pattern to create.

This is a required option.

-layer *layer*

Specifies the layer on which to create the shape_pattern. You must specify a single layer using either the layer name from the technology file or a collection containing exactly one layer.

This is a required option.

-net *net*

Specifies the power/ground net that owns the shapes of this shape_pattern.

The shape_pattern rule_type attribute is automatically set to "none" by default as the owner is power or ground net.

This is a required option for shape_patterns with shape_pattern_type "rect_pattern", "polygon_pattern" or "path_pattern".

-shape_use *use*

Specifies the usage of the shapes in this pattern.

Valid values are

- **area_fill**
  A fill shape that does not require optical proximity correction (OPC). Fill shapes are created inside the top level **fill_cell** for the design, unless a **fill_cell** is specified using **-fill_cell** option.
- **core_wire**
  A shape that connects the endpoints of a follow-pin to its target.
- **detail_route**
  A shape used for detail routed signal nets.
- **follow_pin**
  A shape that connects a standard cell to a power structure.
- **global_route**

A shape used for global routed signal nets.

- **lib_cell_pin_connect**
  A shape that connects to a pin of a standard cell.
- **macro_pin_connect**
  A shape that connects to a pin of an I/O pad cell or macro cell.
- **opc**
  A fill shape that requires optical proximity correction (OPC).
- **pg_augmentation**
  A shape added post route to the power and ground structures.
- **ring**
  A shape that belongs to a power ring.
- **shield_route**
  A shape used for shielding route shapes.
- **stripe** (the default)
  A shape that belongs to a power strap.
- **user_route**
  A shape used for user-entered route shapes.
- **zero_skew**
  A shape used for zero-skew route shapes.

This option is valid only for shape_patterns with shape_pattern_type "rect_pattern", "polygon_pattern" or "path_pattern".

-base_boundary *list_of_points*

Specifies the boundary of a rectangle, polygon or path type base shape. The boundary may be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates (i.e., {*llx lly*} {*urx ury*}). A polygon is specified by its points (i.e., {*x y*} {*x y*} {*x y*} {*x y*}...).

Polygons may also be specified as the combined area of a heterogeneous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

When specifying the boundary polygon as a collection of physical objects, the resulting area must resolve to a single, connected polygon. It is an error to specify a collection of objects with a combined area that resolves to multiple polygons.

This option is required for rectangle and polygon shapes.

-base_path {{*x y*} {*x y*} ...}

Specifies the centerline points of a path type base shape. You must specify at least two points.

If you use this option with other shape types, the tool issues an error.

-base_width *width*

Specifies the width of a path type base shape. The value must be non-negative.

If you use this option with other shape types, the tool issues an error.

-base_start_endcap *endcap*

Specifies the type of the start endcap of a path type base shape.

The valid values are

- **flush** (the default)
  Creates an endcap with square ends and no extension.
- **full_width**
  Creates an endcap with square ends that is extended by the path width.
- **half_width**
  Creates an endcap with square ends that is extended by half the path width.
- **octagon**
  Creates an endcap with octagonal ends and no extension.
- **variable**
  Creates an endcap with square ends that is extended by the value specified in the **-base_start_extension** option. When you specify this setting, the **-base_start_extension** option is required.

This option is valid only for path shapes; the tool issues an error if you use it for other shape types.

-base_end_endcap ***endcap***

Specifies the type of the end endcap of a path type base shape.

The valid values are

- **flush** (the default)
  Creates an endcap with square ends and no extension.
- **full_width**
  Creates an endcap with square ends that is extended by the path width.
- **half_width**
  Creates an endcap with square ends that is extended by half the path width.
- **octagon**
  Creates an endcap with octagonal ends and no extension.
- **variable**
  Creates an endcap with square ends that is extended by the value specified in the **-base_start_extension** option. When you specify this setting, the **-base_start_extension** option is required.

This option is valid only for path shapes; the tool issues an error if you use it for other shape types.

-base_start_extension ***extension***

Specifies the start extension of a path type base shape with variable endcaps. The value must be non-negative.

This option is required when you use the **-base_start_endcap variable** option. If you use this option with other shape types, the tool issues an error.

-base_end_extension ***extension***

Specifies the end extension of a path type base shape with variable endcaps. The value must be non-negative.

This option is required when you use the **-base_end_endcap variable** option. If you use this option with other shape types, the tool issues an error.

-m_dimension ***dimension***

Specifies the number of repetitions in first dimension. The value must be greater than zero. If this option is not specified, the default value is 1.

At least one of *-m_dimension* or *-n_dimension* option must be specified with value greater than 1 to have multiple shapes in the repetition.

-n_dimension *dimension*

Specifies the number of repetitions in second dimension. The value must be greater than zero. If this option is not specified, the default value is 1.

At least one of *-m_dimension* or *-n_dimension* option must be specified with value greater than 1 to have multiple shapes in the repetition.

-m_displacement *delta*

Specifies the uniform offset in first dimension.

This option cannot be specified when *m_dimension* is 1.

This option is mutually exclusive with *-displacements* option. At least one of *-m_displacement*, *-n_displacement* or *-displacements* option must be specified, but all these options cannot be specified together.

-n_displacement *delta*

Specifies the uniform offset in second dimension.

This option cannot be specified when *n_dimension* is 1.

This option is mutually exclusive with *-displacements* option. At least one of *-m_displacement*, *-n_displacement* or *-displacements* option must be specified, but all these options cannot be specified together.

-displacements *delta_list*

Specifies the list of arbitrary offsets for a single dimension.

This option is mutually exclusive with *-m_displacement* and *-n_displacement* options. At least one of *-m_displacement*, *-n_displacement* or *-displacements* option must be specified, but all these options cannot be specified together.

Exactly {m_dimension - 1} or {n_dimension - 1} offsets must be specified for the single dimension {M x 1} or {1 x N} shape_pattern.

-grid_size *gridSize*

Specifies the offset multiplier. The value must be greater than zero. If this option is not specified, the default value is 1.

-occupancy_pattern *occupancy_pattern*

Specifies the occupancy matrix of the shape_pattern. The occupancy_pattern defines a MxN bit pattern of base shape existence. If the occupancy_pattern is smaller than the shape_pattern size, it is replicated in the mDimension-major order. Excess bits in the occupancy_pattern are ignored. The occupancy_pattern is formatted as a space delimited string of equal length substrings (e.g., "1010 0101 1011" is a 3x4 pattern). Each substring must consist of only 1s and 0s, representing a mDimension in the MxN shape_pattern, starting with the first mDimension in the shape_pattern. A 1 means the base-shape is included. A 0 means the base-shape is omitted. If not specified, all the base-shapes are included.

-mask_pattern *mask_pattern*

Specifies the mask-pattern. If the specified value is any one of *alternating*, *alternate_n_dimensions* or *alternate_m_dimensions* then the *-alternating_mask_constraints* option must be specified.

-uniform_mask_constraint *uniform_mask*

Specifies the uniform mask and mask-fixed to use for uniform mask-pattern.

This option is mutually exclusive with *-alternating_mask_constraints* option.

This option may be used when the -mask_pattern option is specified as *uniform*. The default uniform mask is mask_none and it is not fixed.

-alternating_mask_constraints *mask_list*

Specifies the mDimension-major ordered list of masks to use for *alternating*, *alternate_n_dimensions* or *alternate_m_dimensions* mask-pattern.

This option is mutually exclusive with *-uniform_mask_constraint* option.

This option is required when any one of *alternating*, *alternate_n_dimensions* or *alternate_m_dimensions* is specified as the *-mask_pattern* option.

-is_uniform_mask_fixed *true* | *false*

Specifies the uniform mask-fixed to use for all shapes in this shape_pattern.

This option is mutually exclusive with *-uniform_mask_constraint* option.

The default uniform mask is not fixed.

DESCRIPTION

This command creates a new shape_pattern with rectangle, polygon or path type base shape and returns a collection that contains the new shape_pattern.

For a shape_pattern, you must specify the layer (**-layer**) and ownership (**-net**) to a power/ground net.

For rectangle type base shape (**-shape_pattern_type rect_pattern**) and polygon type base shape (**-shape_pattern_type polygon_pattern**), you must specify the boundary (**-base_boundary**). You can also specify the use (**-shape_use**) for the shapes of this shape_pattern.

For path type base shape (**-shape_type path_pattern**), you must specify either combination of path (**-base_path**) and width (**-base_width**) or boundary (**-base_boundary**) with points that form rectangle. You can also specify the use (**-shape_use**), start endcap (**-base_start_endcap**), and end endcap (**-base_end_endcap**) for the shapes of this shape_pattern. If you specify the **-base_start_endcap variable** option, you must also specify the **-base_start_extension** option. If you specify the **-base_end_endcap variable** option, you must also specify the **-base_end_extension** option.

EXAMPLES

The following example creates a 5x4 shape_pattern to repeat the base rectangle {{0 0} {100 100}} on PG net named VDD, but omits the rectanges on the third nDimension of each mDimension.

```
prompt> create_shape_pattern \
        -shape_pattern_type rect_pattern -layer METAL1 \
        -base_boundary {{0 0} {100 100}} -net VDD \
        -m_dimension 5 -n_dimension 4 \
        -m_displacement 150:east -n_displacement 250:north \
        -occupancy_pattern "1101" \
        -mask_pattern alternate_m_dimensions \
        -alternating_mask_constraints \
                {mask_two mask_one mask_three}
```

The following example creates a 3x1 shape_pattern to repeat the base polygon {{0 0} {0 100} {50 100} {50 0}} on PG net named VDD.

```
prompt> create_shape_pattern \
        -shape_pattern_type polygon_pattern -layer METAL1 \
        -base_boundary {{0 0} {0 100} {50 100} {50 0}} \
        -net VDD -m_dimension 3 \
        -displacements {{200:east 300:north} 200:east} \
        -mask_pattern alternating \
        -alternating_mask_constraints \
                {mask_one mask_three mask_two}
```

SEE ALSO

get_shape_patterns(2)
remove_shape_patterns(2)

create_shaping_blockage

Creates a new shaping blockage.

SYNTAX

collection **create_shaping_blockage**
-boundary { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects
[-purpose user | system]
[-name *blockage_name*]
[-cell *cell*]

Data Types

llx	float
lly	float
urx	float
ury	float
x	float
y	float
blockage_name	string
cell	string or collection

ARGUMENTS

-boundary { {{*llx lly*} {*urx ury*}} | {{*x y*} {*x y*} {*x y*} {*x y*}...} } | geometric_objects

Specifies the boundary coordinates of the shaping blockage. The boundary may be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates, for example, {*llx lly*} {*urx ury*}. A polygon is specified by its points, for example, {*x y*} {*x y*} {*x y*} {*x y*}, and so on.

Polygons may also be specified as the combined area of a heterogeneous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

When specifying the boundary polygon as a collection of physical objects, the resulting area must resolve to a single, connected polygon. It is an error to specify a collection of objects with a combined area that resolves to multiple polygons.

This is a required option.

-purpose user | system

Specifies the purpose of the blockage to be created.

Valid values are:

- **user** This should be used by all user-created blockages and ensures that any automated tools will not remove the blockage. This is the default.
- **system** A system blockage can be removed by tools that create and manipulate blockages. You should only create a blockage with this purpose when it may be removed by an automatic tool.

-name *blockage_name*

Specifies the optional name of the blockage. If you specify a name for the blockage, you can get the blockage by the name later in the flow.

-cell *cell*

Specifies the physical cell where the shaping blockage is to be added. The shaping blockage is created in the reference block for the cell by using the coordinate system of the cell argument's top block. The cell must reference a block, not a library cell, unless this command is executed in the library manager. In the library manager, shaping blockages may be created in library cells. When not specified, the shaping blockage is created in the current block.

DESCRIPTION

The **create_shaping_blockage** command creates a shaping blockage. This blockage directs the block shaping commands to avoid areas covered by the blockage.

You can create either rectangular shaping blockages or rectilinear shaping blockages by using the **-boundary** option.

When the tool creates shaping blockages, it automatically assigns a name of SB_*object_id* to each shaping blockage.

EXAMPLES

The following example creates a rectangular shaping blockage with its lower-left corner at (0,0) and its upper-right corner at (100,100).

```
prompt> create_shaping_blockage -boundary {{0 0} {100 100}}
{SB_29440}
```

The following example creates a rectilinear shaping blockage with boundary points at (0,0), (50,0), (50,50), (100,50), (100,100), and (0,100).

```
prompt> create_shaping_blockage \
  -boundary {{0 0} {50 0} {50 50} {100 50} {100 100} {0 100}}
{SB_29696}
```

SEE ALSO

get_shaping_blockages(2)
remove_shaping_blockages(2)

create_shaping_channel

Creates a single channel in the current block.

SYNTAX

channel **create_shaping_channel**
[-left_min *distance*]
[-left_max *distance*]
[-bottom_min *distance*]
[-bottom_max *distance*]
[-right_min *distance*]
[-right_max *distance*]
[-top_min *distance*]
[-top_max *distance*]
[-neighbor *object*]

Data Types

distance float
object string or collection

ARGUMENTS

-left_min *distance*

Specifies the minimum channel spacing along the left side of an object.

-left_max *distance*

Specifies the maximum channel spacing along the left side of an object.

-bottom_min *distance*

Specifies the minimum channel spacing along the bottom of an object.

-bottom_max *distance*

Specifies the maximum channel spacing along the bottom of an object.

-right_min *distance*

Specifies the minimum channel spacing along the right side of an object.

-right_max *distance*

Specifies the maximum channel spacing along the right side of an object.

-top_min ***distance***

Specifies the minimum channel spacing along the top of an object.

-top_max ***distance***

Specifies the maximum channel spacing along the top of an object.

-neighbor ***object***

Specifies a shapeable object relative to which the given channel spacing applies.

DESCRIPTION

The command creates a single channel object in the current block with the specified channel spacing. A channel object must have at least one valid spacing value, so you must specify at least one of the **-left_min**, **-left_max**, **-bottom_min**, **-bottom_max**, **-right_min**, **-right_max**, **-top_min**, or **-top_max** options.

Any number of generic and neighbor channel objects may exist, so this command always creates a new channel object in the block (until the channel capacity in the block is reached).

When you create a neighbor channel by using the **-neighbor** option, you assign the channel constraint to an object by specifying the **create_shaping_constraint -neighbor_channel** command. Channels created without the **-neighbor** option are assigned to an object by specifying the **create_shaping_constraint** command without the **-neighbor_channel** option.

EXAMPLES

The following command creates a generic channel object.

```
prompt> create_shaping_channel -left_max 4
```

The following command creates a neighbor channel object.

```
prompt> create_shaping_channel -top_min 2 -top_max 4 -neighbor my_neighbor
```

The following example creates channel constraints for a boundary channel and a neighbor channel. The boundary channel has a minimum width of 50 microns. The **create_shaping_constraint -boundary_channel** command applies this constraint to the TOP block. The neighbor channel has a minimum width of 200 microns and a maximum width of 250 microns on the top, bottom, left, and right sides of block u3. The **create_shaping_constraint -neighbor_channel** command is used to apply this constraint to the u1 block; any other blocks do not honor this constraint.

```
prompt> set boundary_channel_50_100 \
      [create_shaping_channel -left_min 50 -right_min 50 \
                  -top_min 50 -bottom_min 50]
prompt> create_shaping_constraint [get_block TOP] -type boundary_channel \
  -object_channel $boundary_channel_50_100

prompt> set neighbor_channel [create_shaping_channel -left_min 200 -left_max 250 \
  -right_min 200 -right_max 250 -top_min 200 -top_max 250 \
  -bottom_min 200 -bottom_max 250 -neighbor [get_cells u3]]
prompt> create_shaping_constraint [get_cells {u1}] \
  -type external_channels -neighbor_channel $neighbor_channel
```

SEE ALSO

remove_shaping_channels(2)
get_shaping_channels(2)
channel(3)

create_shaping_constraint

Creates a single shaping_constraint object of a given type.

SYNTAX

shaping_constraint **create_shaping_constraints**
-type *type*
[-object_channel *channel*]
[-neighbor_channels *channels*]
[-relative_x *coord*]
[-relative_y *coord*]
[-min_ratio *ratio*]
[-target_ratio *ratio*]
[-max_ratio *ratio*]
[-min_util *util*]
[-target_util *util*]
[-max_util *util*]
[-min_area *area*]
[-target_area *area*]
[-max_area *area*]
[-reference_object *object*]
[-array_layout north | east | south | west]
[-allowable_orientation set of R0, R180, MX, MY]
[-is_rigid_boundary false | true]
object

Data Types

type string
channel collection
channels collection
coord float
ratio float
util float
area float
object collection

ARGUMENTS

-type *type*

Specifies the type of shaping_constraint object to create. Valid values are **external_channels**, **child_default_channels**, **boundary_channels**, **target_location**, **reference_region**, **alignment_point**, **parent_object**, **aspect_ratio**, **boundary_area**, **child_default_utilization**, **utilization**, **array_layout**, **allowable_orientation**, and **is_rigid_boundary**.

-object_channel *channel*

Specifies the channel object to assign as the generic object channel for channels of type boundary_channels, child_default_channels, and external_channels.

-neighbor_channels *channels*

Specifies the set of channel objects to assign as the neighbor channels for channel valued constraints.

-relative_x *coord*

Specifies the x-coordinate for relative location constraints. Relative location constraints use a float valued pair with values between 0 and 1 to describe a position with the bounding box of a given object. A relative location of {0 0} corresponds to the lower-left bounding box corner, and a relative location of {1 1} corresponds to the upper-right bounding box corner.

-relative_y *coord*

Specifies the y-coordinate for relative location constraints. Relative location constraints use a float valued pair with values between 0 and 1 to describe a position with the bounding box of a given object. A relative location of {0 0} corresponds to the lower-left bounding box corner, and a relative location of {1 1} corresponds to the upper-right bounding box corner.

-min_ratio *ratio*

Specifies the desired minimum value for aspect ratio shaping constraints.

-target_ratio *ratio*

Specifies the desired target value for aspect ratio shaping constraints.

-max_ratio *ratio*

Specifies the desired maximum value for aspect ratio shaping constraints.

-min_util *util*

Specifies the desired minimum value for utilization constraints.

-target_util *util*

Specifies the desired target value for utilization constraints.

-max_util *util*

Specifies the desired maximum value for utilization constraints.

-min_area *area*

Specifies the desired minimum value for area constraints.

-target_area *area*

Specifies the desired target value for area constraints.

-max_area *area*

Specifies the desired maximum value for area constraints.

-reference_object *object*

Specifies one of a voltage area, move bound, shaping group, or block for object valued constraints.

-array_layout north | **east** | **south** | **west**

Specifies the layout direction of objects within the constrained object's boundary.

- north : Objects are arranged from bottom to top
- east : Objects are arranged from left to right
- south : Objects are arranged from top to bottom
- west : Objects are arranged from right to left

-allowable_orientation set of R0, R180, MX, MY

Specifies the allowed orientations of the constrained object. The constraint can be any subset of the orientation values R0, R180, MX, and MY.

-is_rigid_boundary false | true

Specifies the rigidity of the constrained object's boundary. The boundary_status of a constrained object derives from this value as well as the object's physical status.

object

The object for which to create the shaping constraint.

DESCRIPTION

The **create_shaping_constraint** command creates a single shaping constraint of a given type for a given constrainable object. A constrainable object can be a voltage area, move bound, cell, shaping group, or block. Each constrainable object has a certain subset of shaping constraint types that apply to it. For each applicable type, exactly one shaping_constraint object of that type can exist for the given object. Therefore, the **create_shaping_constraint** command either creates a new constraint if one did not previously exist or replaces an existing constraint of the same type.

Each shaping constraint type has a different data value, so the option chosen for the **-type** argument impacts which other arguments are necessary and valid. For example, the parent_object type is object valued, and thus specifying **-type parent_object** requires that you also provide a **-reference_object** argument and restricts you from giving any other data arguments. A complete list of which data arguments are valid for which types is given below. The types are grouped by data value.

Type	Valid options for this type
allowable_orientation	-allowable_orientation
alignment_point target_location	-relative_x, -relative_y
array_layout	-array_layout
aspect_ratio	-min_ratio, -target_ratio, -max_ratio
child_default_utilization utilization	-min_util, -target_util, -max_util
boundary_area	-min_area, -target_area, -max_area
boundary_channels	-object_channel
child_default_channels	-object_channel, -neighbor_channels

external_channels

is_rigid_boundary -is_rigid_boundary

parent_object -reference_object
reference_region

EXAMPLES

The following command creates an alignment_point shaping constraint that applies to a move bound.

```
prompt> create_shaping_constraint -type alignment_point \
  -relative_x 0 -relative_y 0 my_move_bound
```

The following example creates a block shaping constraint. Cells u1, u2, u3, u4, and u5 are grouped into the LEFT_COLUMN group, arranged into a column and placed from bottom to top. Cells u6, u7, u8, u9, and u10 are grouped into the RIGHT_COLUMN group, arranged into a column and placed from top to bottom. The two groups are placed side-by-side with the LEFT_COLUMN group on the left and the RIGHT_COLUMN group on the right.

```
prompt> create_group -name LEFT_COLUMN -shaping [get_cells {u1 u2 u3 u4 u5}]
prompt> create_group -name RIGHT_COLUMN -shaping [get_cells {u6 u7 u8 u9 u10}]
prompt> create_group -name LEFT_AND_RIGHT \
  -shaping [get_groups -shaping {LEFT_COLUMN RIGHT_COLUMN}]
prompt> create_shaping_constraint [get_groups -shaping LEFT_COLUMN] \
  -type array_layout -array_layout north
prompt> create_shaping_constraint [get_groups -shaping RIGHT_COLUMN] \
  -type array_layout -array_layout south
prompt> create_shaping_constraint [get_groups -shaping LEFT_AND_RIGHT ] \
  -type array_layout -array_layout east
prompt> shape_blocks
```

SEE ALSO

create_group(2)
create_shaping_channel(2)
get_shaping_constraints(2)
remove_shaping_constraints(2)
shaping_constraint(3)

create_shields

Performs automatic shield routing.

SYNTAX

status **create_shields**
[-shielding_mode new | unshield | reshield]
[-nets *collection_of_nets*]
[-with_ground *net_name*]
[-preferred_direction_only true | false]
[-align_to_shape_end true | false]
[-ignore_shielding_net_pins true | false]
[-ignore_shielding_net_rails true | false]
[-coaxial_skip_tracks_on_layers {*layer_name_and_number_of_tracks_pairs*}
| [[-coaxial_below true | false]
[-coaxial_above true | false]
[-coaxial_below_skip_tracks *number_of_tracks*]
[-coaxial_above_skip_tracks *number_of_tracks*]
[-coaxial_below_user_spacing *spacing*]
[-coaxial_above_user_spacing *spacing*]]]
[-pg_via_tie_effort_level low | medium | high]
[-shield_via_tie_effort_level low | medium | high]
[-include_adjacent_layers true | false]

Data Types

net_name string
collection_of_nets collection
number_of_tracks int
layer_name_and_number_of_tracks_pairs list

ARGUMENTS

-shielding_mode new | unshield | reshield

Controls which action to perform during automatic shielding.

The definition for each mode is

- **new** (default)
  Creates new shielding for unshielded nets. If a net is previously shielded, even partially, no new shielding is created for that net.
- **unshield**
  Removes existing shielding wires that are associated with shielded nets. All tie-off connections from deleted shielding to the

power and ground network are also deleted to prevent dangling power or ground (PG) nets. The remaining shielding is reconnected to the power and ground network, if needed.

This command can remove only those shielding wires that were created by using the **create_shields** command. Shielding wires created by other means, such as a third-party tool, might not have full association information and must be removed by using the **remove_routes -shield_route** command.

If the tool cannot find any shielding wires associated with shielded nets, no action is performed.

In unshield mode, only the **-nets** option is considered; all other options are ignored.

- **reshield**
  Creates new shielding after unshielding the nets. The shielding is first removed by using the process described for the **unshield** mode. After moving any existing shielding, new shielding is added for the specified nets that do not have shielding.

  Options which used to create shielding will be automatically stored during shield creation and used during reshielding.

  If the design does not contain existing shielding, it is more efficient to use new mode, rather than reshield mode, because in reshield mode the tool first checks for existing shielding and then creates new shielding.

-nets ***collection_of_nets***

Specifies the nets to be shielded, unshielded, or reshielded, depending on the specified mode.

When creating shielding, if the specified nets do not have shielding rules defined, the tool uses the default width and minimum spacing for the layer as defined in the technology file.

By default, this command works on all nets with defined shielding rules. To define the shielding rules, use the **create_routing_rule** and **set_routing_rule** commands.

-with_ground ***net_name***

Specifies which power or ground net to tie the shielding wires to.

By default, the shielding wires are tied to the ground net. If the design contains multiple ground nets, you must use this option to specify the ground net.

-preferred_direction_only true | false

Controls whether the shielding wires are created in the preferred layer direction only.

By default (false), this command creates shielding wires in both preferred and nonpreferred directions, which surround the shielded routing shape.

If this option is true and the shielding wires are created only in the preferred layer direction, the shielding wires might not be connected to each other, in which case more effort is needed to connect these shielding wires to the power and ground nets. Any shielding wires that are not connected to a power or ground net (floating shielding) are deleted. Note that for shielding wires, router does not honor route guides to switch the preferred layer direction.

-align_to_shape_end true | false

Controls whether the shielding wires are aligned to the shielded routing wire ends.

By default (false), this command creates shielding wires that surround the shielded routing shape.

If this option is true, the shielding wires are cut at the shielded wire end points, which might create shielding wires that are not connected to each other, in which case more effort is needed to connect these shielding wires to the power and ground nets. Any shielding wires that are not connected to a power or ground net (floating shielding) are deleted.

-ignore_shielding_net_pins true | false

Controls whether the shielding wires are connected to the standard cell power or ground (PG) pins.

By default (false), this command connects the shielding wires to the standard cell PG pins.

-ignore_shielding_net_rails true | **false**

Controls whether the shielding wires are connected to the standard cell rails.

By default (false), this command connects the shielding wires to the standard cell rails.

-coaxial_skip_tracks_on_layers {*layer_name_and_number_of_tracks_pairs*}

Controls the metal layers on which to perform coaxial shielding for shielded nets and the number of tracks left open between coaxial shielding segments on the specified layers.

You specify the metal layers by using the layer name from the technology file. The number of tracks is an integer value between -1 and 7 that specifies the number of tracks to leave open between coaxial shielding segments.

If the number of tracks is -1 for a layer (the default), this command does not create coaxial shielding on that layer. Otherwise, it skips the specified number of tracks between coaxial shielding segements on that layer.

You cannot combine this option with the non-layer-based method for specifying coaxial shielding (the **-coaxial_below**, **-coaxial_above**, **-coaxial_below_skip_tracks**, **-coaxial_above_skip_tracks**, **-coaxial_below_user_spacing**, and **-coaxial_above_user_spacing** options).

-coaxial_below true | **false**

Controls whether coaxial shielding is created below the shielded net segment layer.

By default (false), this command does not perform coaxial shielding below the shielded net segment layer.

You cannot combine this option with the layer-based method for specifying coaxial shielding (the **-coaxial_skip_tracks_on_layers** option).

-coaxial_above true | **false**

Controls whether coaxial shielding is created above the shielded net segment layer.

By default (false), this command does not perform coaxial shielding above the shielded net segment layer.

You cannot combine this option with the layer-based method for specifying coaxial shielding (the **-coaxial_skip_tracks_on_layers** option).

-coaxial_below_skip_tracks *number_of_tracks*

Controls the number of tracks to be left open between coaxial shielding segments created below the shielded net segment layer.

You can specify an integer value between 0 and 7. The default is 1, which spaces the coaxial shielding segments such that they leave at least one signal routing resource on the layer. To disallow signal routing below the shielded net segment layer, specify 0.

This option is used only when **-coaxial_below** is true.

You cannot combine this option with the layer-based method for specifying coaxial shielding (the **-coaxial_skip_tracks_on_layers** and **-coaxial_below_user_spacing** options).

-coaxial_above_skip_tracks *number_of_tracks*

Controls the number of tracks to be left open between coaxial shielding segments created above the shielded net segment layer.

You can specify an integer value between 0 and 7. The default is 1, which spaces the coaxial shielding segments such that they leave at least one signal routing resource on the layer. To disallow signal routing above the shielded net segment layer, specify

0.

This option is used only when **-coaxial_above** is true.

You cannot combine this option with the layer-based method for specifying coaxial shielding (the **-coaxial_skip_tracks_on_layers** and **-coaxial_above_user_spacing** options).

-coaxial_below_user_spacing ***spacing***

Controls the spacing between coaxial shielding segments created below the shielded net segment layer.

This option is used only when **-coaxial_below** is true. This option doesn't support incremental shielding, this setting will be ignored during incremental shielding.

You cannot combine this option with the layer-based method for specifying coaxial shielding (the **-coaxial_skip_tracks_on_layers** and **coaxial_below_skip_tracks** options).

-coaxial_above_user_spacing ***spacing***

Controls the spacing between coaxial shielding segments created above the shielded net segment layer.

This option is used only when **-coaxial_above** is true. This option doesn't support incremental shielding, this setting will be ignored during incremental shielding.

You cannot combine this option with the layer-based method for specifying coaxial shielding (the **-coaxial_skip_tracks_on_layers** and **coaxial_above_skip_tracks** options).

-include_adjacent_layers true | **false**

Controls whether the shielding wires are created on adjacent (above and below) layers of the shielded net segment layer. This command creates shielding wires with the same preferred direction as the shielded net segments, therefore, the created shielding wires may be in non-preferred layer direction.

By default (false), this command creates same-layer shielding only and does not create shielding wires on adjacent (above and below) layers of the shielded net segment layer.

This option doesn't support incremental shielding, this setting will be ignored during incremental shielding.

-pg_via_tie_effort_level low | **medium** | **high**

Specifies the effort level used to tie the shielding wires to the power and ground network with vias. This option affects the via connections from the shielding wires to PG pins and wires (straps, rails, and rings) that cross shielding wires on adjacent layers.

Valid values for this option are:

- **low**
  Creates a single via or via array connection regardless of the number of intersections between the shielding structure and a PG net on an adjacent layer. Use low effort to speed up the command.
- **medium** (default)
  Creates a via or via array for each intersection between an unconnected wire in the shielding structure and a PG net on an adjacent layer; however, it avoids creating vias or via arrays in the immediate vicinity of already connected shielding wires.
- **high**
  Creates a via or via array connection for each intersection between a shielding wire and a PG net on an adjacent layer. There is a runtime cost associated with using high effort.

-shield_via_tie_effort_level low | **medium** | **high**

Specifies the effort level used to tie the shielding wires to each other with vias. This option affects the via connections between the shielding wires that cross on adjacent layers.

Valid values for this option are:

- **low**
  Does not create via connections between intersecting shielding wires. Use low effort if your design has an enormous number of intersecting shielding wires on adjacent layers, such as when you shield bus nets, which creates a fine grid on adjacent layers. To ensure a high shielding ratio, you should provide a structured power and ground mesh to connect shielding wires to.
- **medium**
  Creates a via or via array connection for each intersection between shielding wires that are associated with the same shielded net.
- **high** (default)
  Creates a via or via array connection for each intersection between shielding wires.

DESCRIPTION

The **create_shields** command shields nets based on the associated shielding rules.

By default, the command performs same-layer shielding on all nets with predefined shielding rules. The router routes shielding wires based on the shielding widths and spacing defined in the shielding rules. The shielding wires are tied to the ground net, standard cell ground pins, and standard cell rails.

To explicitly specify the nets on which to perform shielding, use the **-nets** option. To tie the shielding wires to a named power or ground net, use the **-with_ground** option. To prevent connections to the standard cell PG pins, use the **-ignore_shielding_net_pins** option. To prevent connections to the standard cell rails, use the **-ignore_shielding_net_rails** option.

By default, the command does not perform coaxial shielding. To perform coaxial shielding above the shielded net segment layer, use the **-coaxial_above true** option. To perform coaxial shielding below the shielded net segment layer, use the **-coaxial_below true** option. When the tool performs coaxial shielding, the coaxial shielding segments

- Are routed in the preferred layer direction
- Use the defaultWidth layer attribute from the technology file.
- Are spaced as specified by the **-coaxial_above_skip_tracks** option for coaxial shielding above the shielded net segment layer and as specified by the **-coaxial_below_skip_tracks** option for coaxial shielding below the shielded net segment layer

To remove existing shielding on specified nets, use the **-shielding_mode unshield** option. To remove existing shielding on the whole design, you should use the **remove_routes -shield_route** command.

Prerequisites

Before you run this command, you must first define the shielding rules with the **create_routing_rule** command and assign these rules to the nets to be shielded with the **set_routing_rule** command.

This requirement applies when you do not specify the **-nets** option, you want to use a nondefault width or minimum spacing, you want to snap the shielding segments to the grid, or you want to reserve space for future postroute shielding during signal routing.

Limitations

There might be a discrepancy between DRC violations reported by the **create_shields** and **check_routes** commands. The **create_shields** command works only with shielding shapes and therefore reports only violations for shielding shapes and not for other signal net routing. To get an accurate DRC violations report for all nets, run the **check_routes** command after running the **create_shields** command.

In addition, there might be a slight difference in the shield ratios reported by the **create_shields** and **report_shields** commands. This is due to graph connectivity differences between the two commands. When the reported values differ, use the values reported by the **report_shields** command. The difference in reported values is typically less than three percent, but can be up to five percent.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines the shielding rules; assigns these shielding rules to all clock nets; and then shields the CLK_B1 and CLK_B5 clock nets with the VSS ground net, the CLK_B2 clock net with the GND ground net, and the CLK_B3 and CLK_B4 clock nets with the VDD power net. The shielding wires are not connected to the standard cell PG pins. The CLK_B5 net also has coaxial shielding above and below the shielded net segment layers and no signal routing resources are left between the coaxial shielding segments.

```
prompt> create_routing_rule clock_net_shielding \
  -default_reference_rule -taper_distance 0 -snap_to_track \
  -widths { METAL1 0.16 METAL2 0.2 METAL3 0.2 METAL4 0.2 } \
  -spacings { METAL1 0.18 METAL2 0.21 METAL3 0.21 METAL4 0.21 } \
  -shield_width { METAL1 0.16 METAL2 0.2 METAL3 0.2 METAL4 0.2 } \
  -shield_spacing { METAL1 0.18 METAL2 0.21 METAL3 0.21 METAL4 0.21 } \
  -vias { {via1 1X1 NR} {via2 1X1 NR} {via3 1X1 NR} }

prompt> set clock_nets [get_nets -filter "net_type == Clock" -hier]

prompt> set_routing_rule -rule clock_net_shielding $clock_nets

prompt> create_shields -with_ground VSS -nets {CLK_B1} \
  -ignore_shielding_net_pins true

prompt> create_shields -with_ground GND -nets {CLK_B2} \
  -ignore_shielding_net_pins true

prompt> create_shields -with_ground VDD -nets {CLK_B3 CLK_B4} \
  -ignore_shielding_net_pins true

prompt> create_shields -with_ground VSS -nets {CLK_B5} \
  -coaxial_below true -coaxial_below_skip_tracks 0 \
  -coaxial_above true -coaxial_above_skip_tracks 0 \
  -ignore_shielding_net_pins true
```

The following example adds coaxial shielding for the CLK_B5 net on the M5, M6, and M7 layers. It does not skip tracks on the M5 layer but skips one track on the M6 layer and two tracks on the M7 layer. Other metal layers do not have coaxial shielding.

```
prompt> create_shields -nets {CLK_B5} \
  -coaxial_skip_tracks_on_layers {M5 0 M6 1 M7 2}
```

SEE ALSO

create_routing_rule(2)
remove_routes(2)
set_routing_rule(2)

create_site_array

Creates a site array in the current design.

SYNTAX

collection **create_site_array**
[-name *site_array_name*]
[-site *site_names*]
[-boundary { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects]
[-voltage_area *voltage_area*]
[-flip_first_row true | false]
[-flip_alternate_row true | false]
[-transparent true | false]
[-direction horizontal | vertical]
[-x_margin *x_distance*]
[-y_margin *y_distance_list*]
[-core_offset *xy_distance* | {*x_distance y_distance*}]
[-above *site_array*]
[-below *site_array*]
[-top]
[-bottom]
[-default]
[-aligned true | false]
[-inner_margin {*left bottom right top*}]
[-row_cycles *row_cycle_list*]
[-row_offsets *row_offset_list*]

Data Types

site_array_name string
site_name string or list of string
llx float
lly float
urx float
ury float
x float
y float
geometric_objects collection
voltage_area string or collection
x_distance float
y_distance float or list of float
xy_distance float
site_array string or collection
left float
bottom float

right float
top float
row_cycle_list *collection of integers*
row_offset_list *collection of integers*

ARGUMENTS

-name *site_array_name*

Specifies the name of the site array. If not specified, the tool generates a name for the site array.

-site *site_names*

Specifies the site definitions to use when creating the site array. By default, the default site definition is used if available, otherwise an error is issued. User can specify count after ':' of site_name. This will be useful to create hybrid site-rows i.e. site array with multiple site names.

-boundary {{*llx lly*} {*urx ury*}} | {{*x y*} {*x y*} ...} | *geometric_objects*

Specifies the boundary coordinates of the site array. The boundary can be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates, for example, {*llx lly*} {*urx ury*}. A polygon is specified by its points, for example, {*x y*} {*x y*} {*x y*} {*x y*} and so on.

Polygons can also be specified as the combined area of a mixed collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

When specifying the boundary polygon as a collection of physical objects, the resulting area must resolve to a single, connected polygon. It is an error to specify a collection of objects with a combined area that resolves to multiple polygons.

This option is mutually exclusive with **-voltage_area**.

-voltage_area *voltage_area*

Specifies the name or voltage area object based on whose boundary the site array will be created. The site array is associated with this voltage area and is automatically updated when there are changes to this voltage area boundary. When this voltage area is deleted, then the associated site array are also deleted. This option is mutually exclusive with **-boundary** and **-default**.

-flip_first_row *true* | *false*

Specifies whether the first site row of the site array will be flipped or not. By default, this option is true.

-flip_alternate_row *true* | *false*

Specifies whether alternate site rows of the site array will be flipped or not. Along with **-flip_first_row** this can specify whether even numbered rows are flipped or odd numbered. By default, this option is true.

-transparent true | false

Specifies whether a site array is transparent or opaque. When site arrays are stacked together, this option defines how one site array can affect another. An opaque site array affects the effective site row area for all site arrays that are beneath the opaque site array. By default, this option is false and the site array is opaque. Default site array can't be transparent and this option is ignored for default site array.

-direction horizontal | vertical

Specifies whether a site array is horizontal or vertical. By default, the site array is horizontal.

-x_margin *distance*

Specifies the x-offset between sites within a site row of the site array. By default, the *distance* is zero.

-y_margin *distance_list*

Specifies the y-offset between site rows of the site array. By default, the *distance* is zero. For Hybrid site-rows, specifies the list of y-offset corresponding to site name. If list has less elements as compare to site names then y-offset will be zero for trailing site names and if list has more elements as compare to site names then trailing y-offset will be ignored.

-core_offset *xy_distance* | {*x_distance y_distance*}

Creates an offset to the site grid. Use this option to create offsets from the block origin for aligned site arrays and from the lower left of site array bounding box for unaligned site arrays. The offset values are normalized with respect to site width and height. If one value is specified, the value applies to both x- and y-coordinates of reference point. When two values are specified, the first value is x-coordinate and second is the y-coordinate. By default, the offset values are zero.

-above *site_array*

Places the new site array above *site_array*. This option is mutually exclusive with the **-below**, **-top** and **-bottom** options. By default, the site array is always placed at the top, except for the default site array, which is placed at the bottom.

-below *site_array*

Places the new site array below *site_array*. If the default site array is specified as *site_array*, the new site array will be placed above the default site array. This option is mutually exclusive with **-above**, **-top** and **-bottom**. By default, the site array is always placed at the top, except for the default site array, which is placed at the bottom.

-top

Places the new site array at the top. By default, the site array is always placed at the top, except for the default site array, which will be placed at the bottom. This option is mutually exclusive with **-above**, **-below** and **-bottom**.

-bottom

Places the new site array at the bottom. By default, the site array is always placed at the top, except for the default site array, which will be placed at the bottom. If a default site array already exists, the site array will be placed above the default site array. This option is mutually exclusive with the **-above**, **-below** and **-top** options.

-default

Specifies that the newly created site array will be the default site array. This option is useful when default site array must be created with an input boundary. Use this option to specify if the site array created with input boundary should be default or not. This option is mutually exclusive with **-voltage_area**.

-aligned true | false

Specifies the site array to be aligned or unaligned. By default, this option is true. Default site array can't be unaligned and this option is ignored for default site array.

-inner_margin {*left bottom right top*}

Specifies the inner margin to be applied to shape of site array. This is to shrink the shape of site array for effective region of its site rows. It is helpful to maintain a no site row region for site arrays derived from voltage areas.

-row_cycles *row_cycle_list*

Specifies one or more value of cycle for rows of site-array. Cycle will be assigned to site-rows in order as specifed in command

option. Value of cycle on first site-row will depend upon **-aligned** option.

-row_offsets ***row_offset_list***

Specifies one or more value of offset for rows of site-array. Offset will be assigned to site-rows in order as specified in command option. Value of offset on first site-row will depend upon **-aligned** option.

DESCRIPTION

The **create_site_array** command creates a site array with the specified boundary in the current design. Site arrays can be stacked together and their transparency dictates the effective area available for site row creation for each of the stacked site arrays. Some of the properties, such as flipping first site row or alternate site rows, describes how they are going to appear in the design. The site rows of a site array can be made horizontal or vertical. They have a name and are based on a basic site definition or multiple site definition. In case of multiple site definitions i.e. hybrid site-array, User can give count followed by ':' after site name like {unit1:2 unit2:3}. Count will used to create number of site-rows of corresponding site name before site row creation of next site name.

There can be a default site array inside a block. When no reference boundary, either through **-boundary** or **-voltage_area**, is provided, or a boundary with **-boundary** is provided along with **-default** option, then a default site array is created in the current block. The site array tracks the boundary of the block and is placed at the bottom of the stack as an opaque and aligned site array.

The region for the default site array is calculated by shrinking the block boundary by the block hard keepout margin and inner margin if applicable, and taking the common area with any input boundary provided by the user. If the block boundary changes or the hard keepout margin for the block changes, then the site array also changes accordingly.

The nondefault site array considers block hard keepout margins and inner margins if applicable for its effective site rows calculation. If any part of nondefault site array falls within a hard keepout margin region, that portion is removed from any effective site rows calculation. If the block boundary changes or the hard keepout margin for the block changes, then the nondefault site arrays also change.

A site array can be associated with a voltage area. In this case, the site array tracks the boundary of voltage area and changes according to any changes in the voltage area boundary. Removing such a voltage area will cause removal of the associated site array. Since the voltage area can have multiple regions, this will result in multiple site arrays stacked on top of each other.

The site arrays have two flavors, aligned and unaligned. Aligned site arrays have their site rows aligned to a virtual site grid passing through the block's origin plus any offset specified. The **-flip_first_row** value applies to the first site row starting at block's origin plus any offset specified. As a result, the first site row within the site array might be flipped or not, depending on the location of site array on the virtual site grid. Unaligned site arrays have their site rows directly based on site array boundary starting at lower-left corner of site array bounding box, taking into account any offset specified. So first site row within unaligned site array will be based on **-flip_first_row** value.

Cycle and offset will be assigned to site-rows in same order in which they specified but cycle/offset on first site-row will depend upon **-aligned** option. If **-aligned** is false then first site-row will have first value specified as command option **-row_cycles**/**-row_offsets**.

EXAMPLES

The following example creates an unaligned site array named SA1 with a rectangular boundary. SA1 is opaque with horizontal rows. Odd numbered rows are flipped and are at the top of the stack.

```
prompt> create_site_array -name SA1 -site unit -boundary \
  {{100 100} {200 200}} -aligned false
```

The following example creates an aligned site array named SA2 with its boundary copied from the voltage area DEFAULT_VA. SA2

is opaque with vertical rows. The site array is placed above the existing site array SA1. This may cause the effective area of SA1 to change and therefore change in its site rows, since SA2 is opaque.

```
prompt> create_site_array -name SA2 -site unit -voltage_area DEFAULT_VA \
  -above SA1 -direction vertical
```

The following example creates a default site array named SA1 with its boundary as the boundary of the block shrunk by hard keepout margin. When the block boundary changes, or its hard keepout margin changes, the boundary of SA1 (and therefore its site rows) will also change. SA1 is opaque with horizontal rows and will be at the bottom of the stack.

```
prompt> create_site_array -name SA1 -site unit
```

The following example creates a default site array with a system-generated name with a rectangular boundary. When the block boundary changes, or its hard keepout margin changes, the default site array boundary (and therefore its site rows) might also change depending on input boundary. The default site array is opaque with horizontal rows and will be at the bottom of the stack.

```
prompt> create_site_array -boundary {{100 100} {200 200}} -default
```

The following example creates aligned site array named SA1 with its boundary copied from the voltage area VA1 and has an inner margin. Boundary of VA1 will be shrunk by specified inner margin. SA1 is opaque with vertical rows.

```
prompt> set voltage_area VA1
prompt> create_site_array -name SA1 -site unit -voltage_area $voltage_area \
  -inner_margin {10 20 10 20}
```

The following example creates an site array named SA1 with a rectangular boundary. SA1 will have 3 site-rows of site name 'unit' followed by 2 site-rows of site name 'punit'. Similar pattern will follow after that. y-offset between site-rows of site name 'unit' will be 0.5 and y-offset between site-rows of site name 'punit' will be 1.

```
prompt> create_site_array -name SA1 -site {unit:3 punit:2} -boundary \
  {{100 100} {200 200}} -y_margin {0.5 1}
```

SEE ALSO

get_site_arrays(2)
remove_site_arrays(2)

create_site_def

Creates a site def using the technology of the current or the specified library.

SYNTAX

collection **create_site_def**
-name *site_def_name*
-height *distance*
-width *distance*
[-is_default]
[-library *library*]
[-tech *tech_object*]
[-type *type*]
[-symmetry *symmetry*]

Data Types

site_def_name string
distance float
library collection
tech_object collection
type string
symmetry string

ARGUMENTS

-name *site_def_name*

Specifies the name of the site_def to be created.

-height *distance*

Specifies the height of the site_def to be created.

-width *distance*

Specifies the width of the site_def to be created.

-is_default

Specifies the site definition as default. Creates a default site_def for the specified tech unless the default already exists. Results in error and returns NULL if the default site_def of the specified tech already exists.

-library *library*

Site definition will be created in the technology of this library. Default is current library's tech. This is mutually exclusive with **-tech**

option.

-tech ***tech_object***

Site definition will be created in this technology. Default is current library's tech. This is mutually exclusive with **-library** option.

-type ***type***

Specifies the type of the site definition as core/pad. Default is core.

-symmetry ***symmetry***

Specifies if the site definition is symmetrical about X and/or Y and/or 90 degree rotation. By default there is no symmetry.

Valid symmetry is a set or combination of the following values

- **X** (symmetrical about the X-axis)
- **Y** (symmetrical about the Y-axis)
- **R90** (symmetrical for 90 degrees rotation)

DESCRIPTION

The **create_site_def** command enables you to create a site def in the tech of the current or specified library. A tech can have at-most one default site definition. The use of "-is_default" option would result in error if the tech already has a default site definition.

EXAMPLES

The following example creates a core type site def named 'SD1' with height 0.15 and width 0.25 in the tech of current library. SD1 will be symmetrical about both horizontal and vertical axes.

```
prompt> create_site_def -name SD1 -height 0.15 -width 0.25 \
  -symmetry { X Y }
```

The following example creates a pad type site def named 'SD2' with height 1.5 and width 0.15 in the tech of library r4000. SD1 will be symmetrical about X and Y and 90 degree rotation.

```
prompt> create_site_def -library r4000 -name SD2 -height 1.5 \
  -width 0.15 -type pad -symmetry { X Y R90 }
```

SEE ALSO

get_site_defs(2)
report_site_defs(2)
remove_site_defs(2)

create_site_row

Creates a site row in the current design.

SYNTAX

collection **create_site_row**
-name *site_row_name*
-site *site_name*
-origin {*x y*}
-site_count *count*
[-orientation *orientation*]
[-site_orientation *orientation*]
[-x_margin *distance*]

Data Types

site_row_name string
site_name string
x y point
count int
orientation string
distance float

ARGUMENTS

-name *site_row_name*

Specifies the name of the site row.

-site *site_name*

Specifies the name of the site based on which the site row will be created.

-origin {*x y*}

Creates a site row at the specified x- and y-coordinate. The coordinate is relative to the die origin.

-site_count *count*

Specifies the number of sites to be put in the site row. This determines the bbox of the site row and core area of the block.

-orientation *orientation*

Specifies the orientation of a site row. The default orientation is R0.

Valid values are

- **MX** (mirror around the x-axis)
- **MXR90** (mirror around the x-axis and rotate 90 degrees)
- **MY** (mirror around the y-axis)
- **MYR90** (mirror around the y-axis and rotate 90 degrees)
- **R0** (no rotation, which is the default)
- **R90** (rotate 90 degrees)
- **R180** (rotate 180 degrees)
- **R270** (rotate 270 degrees)

-site_orientation *orientation*

Specifies the orientation of sites within a site row. The orientation is relative to the orientation for the site row. By default, the orientation is R0.

Valid values are

- **MX** (mirror around the x-axis)
- **MXR90** (mirror around the x-axis and rotate 90 degrees)
- **MY** (mirror around the y-axis)
- **MYR90** (mirror around the y-axis and rotate 90 degrees)
- **R0** (no rotation, which is the default)
- **R90** (rotate 90 degrees)
- **R180** (rotate 180 degrees)
- **R270** (rotate 270 degrees)

-x_margin *distance*

Specifies the x-offset between sites within a site row. By default, the offset is zero.

DESCRIPTION

The **create_site_row** command creates a site row in the current design, at the specified origin with specified number of sites. The site rows have a name and are based on a basic site definition. The number of sites determine the bounding box of a site row and the core area of the block. The created site row cannot overlap any existing site arrays.

EXAMPLES

The following example creates a site row named SR1 with 1000 sites at location 0,100. SR1 will be horizontal with horizontal sites in it.

```
prompt> create_site_row -name SR1 -site unit -origin \
  {0 100} -site_count 1000
```

The following example creates a site row named SR1 with 1000 sites at location 0,100. SR1 will be vertical with relative horizontal sites in it.

```
prompt> create_site_row -name SR1 -site unit -origin \
  {0 100} -site_count 1000 -orientation R90
```

SEE ALSO

get_site_rows(2)
remove_site_rows(2)

create_stdcell_fillers

Fills empty spaces in standard cell rows with filler cells.

SYNTAX

status **create_stdcell_fillers**
-lib_cells *lib_cells*
[-one_fin_lib_cells *lib_cells*]
[-post_eco]
[-ignore_hard_blockages]
[-prefix *prefix*]
[-bboxes *rectangle*]
[-voltage_area *voltage_areas*]
[-rules *rule_list*]
[-continue_on_error]
[-utilization *float_value*]
[-smallest_cell_size *int_value*]
[-type_utilization *lib_cells_utilization_pairs*]
[-fill_remaining]
[-leakage_vt_order *vt_layer_list*]
[-leakage_vt_check]
[-flip]
[-prefer_type_ordering]

Data Types

lib_cells	collection
prefix	string
rectangle	rectangle
voltage_areas	collection
rule_list	list
float_value	float
int_value	int
lib_cells_utilization_pairs	list
vt_layer_list	list

ARGUMENTS

-lib_cells *lib_cells*

Specifies the filler cells to use.

By default, the tool adds the filler cells in the order that you specify. For best results, specify the cells from largest to smallest; this is a requirement when the no_1x rule or advanced rules are used.

-one_fin_lib_cells *lib_cells*

Specifies the one-fin filler cells to use. The specified cells must not exist in the list specified by the **-lib_cells** option.

The command minimizes the usage of one-fin filler cells; these cells are used only when none of the filler cells specified in the **-lib_cells** option can be legally inserted.

This option applies only when the advanced legalizer is enabled by setting the **place.legalize.enable_advanced_legalizer** application option to **true**.

-post_eco

Marks new filler cells so the **remove_stdcell_fillers_with_violation** command can identify them in the post-ECO flow.

-ignore_hard_blockages

Creates filler cells in areas that have hard placement blockages.

-prefix *prefix*

Adds the specified prefix to the instance names of the filler cells inserted by this command. You can later use the prefix to identify the filler cell instances.

-bboxes *rectangle*

Specifies the rectangular region (in microns) in which to insert filler cells. To specify the rectangular region, use the following syntax to specify the coordinates of the lower-left and upper-right corners of the region:

{ {*llx lly*} {*urx ury*} }

By default, filler cells are inserted throughout the entire chip.

-voltage_area *voltage_areas*

Specifies the voltage areas in which to insert filler cells.

By default, the tool inserts filler cells in all voltage areas.

-rules *rule_list*

Specifies the special rules to follow during filler cell insertion.

You can specify one or more of the following keywords:

- **no_1x**
  Prevents the command from placing filler cells such that it would leave a 1x gap.
- **color_safe**
  Ensures color correctness for multiple-patterning designs. This rule applies only when color-aware placement is enabled by setting the **place.legalize.enable_color_aware_placement** application option to **true**.
- **check_pnet**
  Enables power net checking for the filler cells. By default, power net checking is disabled to save runtime.

-continue_on_error

Skips unfillable cell locations and continues to the next gap.

This option applies only when advanced rules are used. Use this option only when the list of filler cells specified by the **-lib_cells** option does not have all the necessary filler cells to fill all possible gaps. For example, use this option when you are performing decoupling capacitor insertion but do not have smaller decoupling capacitors to fill all possible gaps. In this case, the remaining gaps must be filled during nonmetal filler cell insertion.

When you specify this option, it is assumed that you will fill the remaining cell locations later. The tool avoids trying all possible cell combinations to save runtime and leaves fillable gaps unfilled.

-utilization ***float_value***

Specifies the maximum percentage of free space to fill with filler cells. The default is 100.

-smallest_cell_size ***int_value***

Specifies the smallest cell size for the design, which must be an integer between 1 and 11. The default is 1.

If the design has no 1x cells, use either **-rules {no_1x}** or **-smallest_cell_size 2**. Do not use both.

-type_utilization ***lib_cells_utilization_pairs***

Controls different target area percentages for one or more of the library cells specified in the **-lib_cells** option. Specify the argument as a list of pairs in the following format:

{{*lib_cells*} *utilization* ...}

For example, the following option specifies that total area percentage for the lib_A1 and lib_A2 library cells is 20 percent and the total area percentage for the lib_B1 library cell is 30 percent:

```
-type_utilization { {lib_A1 lib_A2} 20 lib_B1 30 }
```

As with the **-lib_cells** option, the library cells in each pair should be specified in descending order by cell size. You can use the * wildcard character to specify the library cell names. The percentage specification must be greater than 0 and less than or equal to 100. The sum of all the percentages must be less than or equal to 100.

-fill_remaining

Uses the cells specified in the **-lib_cells** option but not in the **-type_utilization** option to fill the design.

When you specify this option, the command fills all empty space. The command fails if you specify this option but all library cells have been specified in the **-type_utilization** option and there are no remaining library cells available to fill the rest of the design.

You can use this option only when the **-type_utilization** option is specified.

-flip

Only works with advanced legalizer. Normally filler insertion will not try flipping when the violation is due to pin color alignement. If you use this option and filler insertion encounter pin color alignment violation, it will try inserting with flipped orientation as well.

-prefer_type_ordering

When this option is specified, it will insert lib cells lists specified in **-type_utilization** one by one from the left to right. User can use this option to improve the real utilization of those lib cells difficult to be inserted due to legalization rule or lib cell size compared with normal type utilization mode. But it may increase the runtime compared with normal type utilization mode. This option must be used together with **-type_utilization** option. Otherwise, it will error out.

This option applies only when the advanced legalizer is enabled by setting the **place.legalize.enable_advanced_legalizer** application option to **true**.

-leakage_vt_order ***vt_layer_list***

Specifies the threshold voltage layers in order of decreasing leakage current. The command uses this information to reduce the leakage current by selecting the filler cells with the lowest leakage current.

The specified threshold voltage layers must include all layers used by the filler cells specified in the **-lib_cells** option.

This option applies only when the advanced legalizer is enabled by setting the **place.legalize.enable_advanced_legalizer** application option to **true**.

-leakage_vt_check

Checks if the current filler cell insertion result has the lowest leakage power when using the filler cells specified in the **-lib_cells** option.

By default, the command reports a maximum of 100 violations. To change the maximum number of reported violations, set the **chf.create_stdcell_fillers.max_leakage_vt_order_violations** application option.

When you use this option, the command ignores the **-bboxes** and **-voltage_areas** options.

You can use this option only when the **-leakage_vt_order** and **-lib_cells** options are specified.

DESCRIPTION

This command fills empty spaces in standard cell rows with instances of reference filler cells from the library. You should perform placement (and clock tree synthesis, if applicable) before adding filler cells.

A typical flow is

1. Use the **create_stdcell_fillers** command to insert decoupling capacitors.
2. Update the PG connections by using the **connect_pg_net -automatic** command.
3. Use the **remove_stdcell_fillers_with_violation** command to remove the decoupling capacitors with violations.
4. Use the **create_stdcell_fillers** command to insert non-metal filler cells.
5. Update the PG connections by using the **connect_pg_net -automatic** command.

After filler cell insertion, use the **check_legality** command to verify the filler cell placement legality.

EXAMPLES

The following example fills empty spaces with filler cells from the mylib reference library named FILL_4X, FILL_2X, and FILL_1X in the specified order. If the sizes of the filler cells are such that FILL_4X > FILL_2X > FILL_1X, this ordering minimizes the number of filler cells added by using the larger filler cells first.

```
prompt> create_stdcell_fillers \
  -lib_cells {mylib/FILL_4X mylib/FILL_2X mylib/FILL_1X}
```

SEE ALSO

check_legality(2)
connect_pg_net(2)
fix_placement_color_mask(2)
remove_stdcell_fillers_with_violation(2)
chf.create_stdcell_fillers.max_leakage_vt_order_violations(3)
place.legalize.enable_advanced_legalizer(3)
place.legalize.enable_color_aware_placement(3)

create_stub_chain

Creates a stub chain in the current design.

SYNTAX

collection **create_stub_chain**
-name *name*
[-partition *name*]
[-max_bits *number_of_bitsl*]
[-startpoint *port_or_pin*]
[-endpoint *port_or_pin*]
[-floating_elements *elements_lists*]
[-ordered_elements *elements_lists*]

Data Types

name string
number_of_bits integer
port_or_pin collection
elements_list list

ARGUMENTS

-name *name*

Name of the scan_chain

-partition *name*

Scan partition name

-max_bits *number_of_bits*

Maximum bit length of the stub_chain

-startpoint *port_or_pin*

Starting physical port or pin

-endpoint *port_or_pin*

Ending physical port or pin

-floating_elements *elements_lists*

Floating element lists to add to the new stub_chain. The scan-in and scan-out pin of each triplet must be from the same scan cell. List Format: {{{ scan_in_pin_name scan_out_pin_name bit_length } ... } ... }

-ordered_elements elements_list

Ordered element lists to add to the new stub_chain. The scan-in and scan-out pin of each triplet must be from the same scan cell. List Format: {{{ scan_in_pin_name scan_out_pin_name bit_length } ... } ... }

DESCRIPTION

This command creates a new stub chain and returns a collection containing the new stub chain.

EXAMPLES

The following example creates a stub chain with port1 as the startpoint and port2 as the endpoint.

```
prompt> create_stub_chain -name stub1 -startpoint port1 -endpoint port2
```

The following example creates a stub chain with floating and ordered_elements.

```
prompt> create_stub_chain -name stub2 -floating_elements {{{u1/pin1 u1/pin2 4} {u1/pin3 u1/pin4}} {{u2/pin1 u2/pin2 4}}}
-ordered_elements {{{u3/pin1 u3/pin2 4}}}
```

SEE ALSO

get_stub_chains(2)
remove_stub_chains(2)

create_supernet

Creates a new supernet.

SYNTAX

collection **create_supernet**
pin_or_port
[-design *design*]
[-name *supernet_name*]
[-start *supernet_start_pin_or_port*]
[-stop *supernet_stop_pin_or_port*]

Data Types

pin_or_port collection
design collection
supernet_name string
supernet_start_pin_or_port pin_port_object
supernet_stop_pin_or_port pin_port_object

ARGUMENTS

pin_or_port

Specifies the anchor object for the supernet. It is an error if a supernet already exists on that anchor.

This is a required argument.

-design *design*

Specifies the block in which to create the supernet.

By default, the supernet is created in the current block.

-name *supernet_name*

Specifies the name of the supernet.

By default, the name is the name of the anchor object, possibly with a numerical postfix to make it unique.

-start *supernet_start_pin_or_port*

Specifies the pin or port which is start anchor of the supernet. It is an error if a supernet already exists on start.

This is a required argument if pin_or_port is not specified.

-stop *supernet_stop_pin_or_port*

Specifies the pin or port which is stop anchor of the supernet.

This is a required argument if pin_or_port is not specified and -start is specified.

DESCRIPTION

This command creates a new supernet with the specified name and anchor object. The new supernet is returned as a collection. If -start and -stop is specified then it will create a supernet between start to stop and return an supernet.

start and stop anchor must be on block-boundaries and both in same physical hierarchy only.

This command marks cells and pins as being transparent for supernets from the design.

The set of nets assigned to a particular supernet are automatically derived by transparently tracing in both directions through repeaters (and cells explicitly specified by the user) from the specified anchor point defined for the supernet. In case of supernet created using start and stop, set of nets are nets from start to stop. If no path exists from start to stop then it will not return any get. If multiple paths exists from start to stop then supernet is set of all the nets between start to stop from all paths.

By default tracing will see through repeaters (buffers and inverters only) only. Non-repeater cells can be marked as transparent for supernet tracing using the set_supernet_exceptions command. Also, user can disable certain repeater cells for supernet tracing by marking them non-transparent using set_supernet_exceptions command. These supernet exceptions have to be defined prior to the creation of supernets.

EXAMPLES

The following example creates a supernet on pin instA/01 and names it IF1.

prompt> **create_supernet [get_pins instA/O1] -name IF1**

The following example creates a supernet between i1 to o1 and names it S1.

prompt> **create_supernet -start i1 -stop o1 -name S1**

SEE ALSO

get_supernets(2)
remove_supernets(2)
set_supernet_exceptions(2)
report_supernet_exceptions(2)

create_supply_net

Creates a supply net for the specified power domain. The supply net is created in the logic hierarchy at the same scope as the specified power domain.

SYNTAX

string **create_supply_net**
[-domain *domain_name*]
[-reuse]
[-resolve unresolved | parallel | one_hot | parallel_one_hot]
supply_net_name

Data Types

domain_name string
supply_net_name string

ARGUMENTS

supply_net_name

Specifies the name of the supply net to be created. The name should be a simple (non-hierarchical) name.

In the scope of specified power domain, if there is a supply net or logical hierarchical net with the same name, the supply net cannot be created. An exception to this rule is when you use the **-reuse** option. In this case the net name must be same as an existing supply net in the same scope; name; check is not performed.

This argument is required.

-domain *domain_name*

Specifies the power domain for which this supply net is defined. The power domain must exist.

This argument is optional. If -domain is not specified, the supply net will be available for all power domains at the current scope and below.

-reuse

Reuses an existing supply net instead of creating a new one. One supply net can be associated with several power domains. If this option is used, the specified supply net must already exist.

-resolve unresolved | parallel | one_hot | parallel_one_hot

Specifies how the state and voltage of the supply net are resolved when the supply net is driven by a power switch or multiple power switches. Specifying *unresolved* means that this supply net allows only a single driver. Specifying *parallel* means that this supply net can be driven by multiple power switches. When the supply net is connected to the output of multiple power switches,

any number of outputs may be ON at the same time, as long as the voltage value is same. Specifying *one_hot* means multiple supply sources, each having a unique driver, may be connected to the supply net. Specifying *parallel_one_hot* allows resolution of a supply net that has multiple root supply drivers where each driver may have more than one path through supply sources to the supply net.

Users can specify their own custom resolution functions, defined during simulation. The resolution function thus defined can be used as the value to the -resolve option. Custom resolutions allows for multiple drivers.

The default is **unresolved**.

DESCRIPTION

The **create_supply_net** command enables you to create a supply net defined in the specified power domain. A supply net presents the intention of the power supply in the power domains. A supply net connects supply ports and, or pins.

Each power domain has a primary power supply net and a primary ground supply net, and could have several other supply nets. If the command succeeds, a supply net is created in the scope of the power domain. The new supply net is neither a primary power net nor a primary ground net of the power domain. Use the **set_domain_supply_net** command to set the supply net as the primary power or ground net.

On success this command returns the full name of the supply net (from the current scope). On failure, it returns a null string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a supply net named A_VDD in the power domain PD1, and recreates supply net A_DD in the power domain PD2 using the **-reuse** option:

```
prompt> create_power_domain PD1 -element INST1
PD1

prompt> create_power_domain PD2 -element INST2
PD2

prompt> create_supply_net A_VDD -domain PD1
A_VDD

prompt> create_supply_net A_VDD -domain PD2 -reuse
A_VDD
```

SEE ALSO

report_supply_nets(2)

create_supply_port

Creates a supply port in the specified power domain. If a power domain is not specified, creates the port in the current scope.

SYNTAX

string **create_supply_port**
supply_port_name
[-domain *domain_name*]
[-direction in | out | inout]

Data Types

supply_port_name string
domain_name string

ARGUMENTS

supply_port_name

This is a required argument.

Specifies the name of the supply port to be created. The name should be a simple (non-hierarchical) name. This argument can be used with the **-domain** option to avoid hierarchical names.

If a supply port with same name exists, this command will result in an error. If a design port with same name exists, this command will continue to create a supply port without any warning, error. Especially, if this existing port is signal type, then it will be treated as a PG (with direct tie with the signal pins), let resolve_pg_net handle it.

-domain *domain_name*

Specifies the power domain where this port defines a supply net connection point. If the specified power domain does not exist in the current scope, the command fails.

-direction in | **out** | **inout**

Defines how the state information is propagated through the supply network when connected to the port. If the port is an input port, the state information of the external supply net connected to the port is propagated into the domain. Similarly, for an output port, the state information of the internal supply net connected to the port is propagated outside the domain.

The default value of this option is **in**.

DESCRIPTION

The *create_supply_port* command creates a supply port at the scope of the specified power domain or at the current scope. A supply port provides a connection point for the supply net.

On success, the command returns the full name of the supply port (from the current scope). Otherwise, returns a null string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates the supply port *VN1* at the scope of power domain *PD1*:

```
prompt> create_power_domain PD1 -element INST1
PD1

prompt> create_supply_port VN1 -domain PD1
VN1
```

SEE ALSO

report_supply_ports(2)

create_supply_set

Creates or updates a set of supplies that can be used to define the power network. A supply set is created in the current logic hierarchy.

SYNTAX

string **create_supply_set**
[-function {function_name supply_net_name}*]
[-update]
supply_set_name

Data Types

supply_set_name string
function_name string
supply_net_name string

ARGUMENTS

supply_set_name

Specifies the name of the supply set to be created. The name should be a simple (non-hierarchical) name.

If a supply set with the same name exists in the current scope, the supply set is not created.

An exception to this rule is when you use the **-update** option. In this case the supply set name must be same as an existing supply set in the same scope.

This argument is required.

-function {*function_name supply_net_name*}*

Specifies the functionality to which an actual supply net named **supply_net_name** should be associated.

This argument takes two parameters as input. First is the name of the functionality, which is either **power** or **ground**. Second parameter is the name of the supply net that is in the same hierarchical scope of the supply set.

This argument is required when the **-update** argument is used. Optional otherwise.

-update

Instructs the tool to associate an actual supply net to the **power** and **ground** functionalities of an existing supply set. It is an error if the supply set specified does not exist. Only one update is allowed for each functionality of the supply set. If your update is not successful, the tool issues an error message. You can continue to update the functionality, until your update is successful.

This argument is optional. When the **-update** argument is used, the **-function** argument must also be used.

DESCRIPTION

The **create_supply_set** command creates a supply set in the current scope. A supply set eliminates the requirement of a supply net and supply ports in the design in the frontend tool. However, actual supply nets must be associated with the supply sets later on in the flow before physical synthesis is done.

A supply set supports the use of the following functions in the design:

a) Power

b) Ground

These functions, when unassociated with any supply net, can be used as supply nets for any purpose. They can be accessed by referencing them through the name of the supply set. To access the **power** function, you must specify *supply_set_name.power* and to access the **ground** function, you must specify *supply_set_name.ground*.

The functions of a supply set can be used just as any other supply net in the UPF design, with no domain restrictions. However, domain restrictions are applicable when an **-update** is called. This means that the actual supply nets, being associated with a supply set, will be checked for their definitions in the domains where the supply set functions have been used.

Calling an update on an existing supply set means replacing the supply set functions with actual supply nets. After doing an **-update**, the actual supply net can be referenced through the supply set or by its actual name.

EXAMPLES

The following example creates a supply set named primary_sset in the current logical hierarchy.

```
prompt> create_supply_set primary_sset
primary_sset
```

The following example illustrates how a supply set function can be used.

```
prompt> create_supply_set primary_sset
primary_sset

prompt> create_power_domain PD_TOP
PD_TOP

prompt> set_domain_supply_net PD_TOP A \
        -primary_power_net primary_sset.power \
        -primary_ground_net primary_sset.ground \
1
prompt> set_retention ret_cstr -domain PD_TOP \
        -retention_power_net primary_sset.power \
        -retention_ground_net primary_sset.ground
1
prompt> set_isolation iso_cstr -domain PD_TOP \
        -isolation_power_net primary_sset.power \
        -isolation_ground_net primary_sset.ground \
1
```

The following example illustrates how a supply set can be associated with actual supply nets.

```
prompt> create_supply_set primary_sset
primary_sset

prompt> create_power_domain PD_TOP
PD_TOP

prompt> set_domain_supply_net PD_TOP \
        -primary_power_net primary_sset.power \
        -primary_ground_net primary_sset.ground
1
prompt> set_retention ret_cstr -domain PD_TOP \
        -retention_power_net primary_sset.power \
        -retention_ground_net primary_sset.ground
1
prompt> set_isolation iso_cstr -domain PD_TOP \
        -isolation_power_net primary_sset.power \
        -isolation_ground_net primary_sset.ground \
1
prompt> create_supply_net TOP_VDD -domain PD_TOP
TOP_VDD
prompt> create_supply_net TOP_VSS -domain PD_TOP
TOP_VSS
prompt> create_supply_set primary_sset \
       -function {power TOP_VDD} \
       -function {ground TOP_VSS} \
       -update
1
```

SEE ALSO

create_supply_net(2)
set_domain_supply_net(2)
set_retention(2)
set_isolation(2)

create_tap_cells

Inserts tap cells in the current block to form a two-dimensional array structure to ensure that all standard cells placed subsequently comply with the maximum diffusion-to-tap distance limit.

SYNTAX

status **create_tap_cells**
-lib_cell *cell_name* | -multiple_tap_cells *table*
[-mirrored_row_lib_cell *cell_name*]
-distance *tap_distance*
[-voltage_area *voltage_areas*]
[-offset *distance*]
[-pattern every_row | every_other_row | stagger]
[-insert_into_blocks]
[-skip_fixed_cells]
[-prefix *cell_prefix*]
[-separator *cell_separator*]
[-at_distance_only]
[-enable_prerouted_net_check]
[-min_horizontal_periphery_spacing *number_of_unit_tiles*]
[-row_end_tap_bypass]
[-preserve_distance_continuity *lib_cells*]
[-min_row_edge_distance_layer_override *layerMinRowPair*]
[-other_cells_with_well_taps *table*]
[-cells_not_breaking_well_continuity *cell_names*]
[-cells_breaking_nwell_continuity *cell_names*]
[-cells_breaking_pwell_continuity *cell_names*]

Data Types

cell_name string
cell_names string
tap_distance float
voltage_areas collection
distance float
cell_prefix string
cell_separator string
number_of_unit_tiles integer
lib_cells collection
table string

ARGUMENTS

-lib_cell *cell_name*

Specifies the library reference cell to be used as a tap cell. You must specify a single library cell. This is a required option.

-mirrored_row_lib_cell *cell_name*

Specifies the library reference cell to be used as a tap cell on mirrored rows. Mirrored rows are rows whose row orientation is set to MX rather than R0. You can only specify a single library cell.

If you do not use this option, the command uses the cell specified with the **-lib_cell** option on the mirrored rows.

-distance *tap_distance*

Specifies the distance in microns between two tap cells in a row. This distance is referred to as the tap distance. This is a required option.

-voltage_area *voltage_areas*

Specifies the voltage areas in which to create tap cells.

By default, the command creates tap cells in all voltage areas.

-offset *distance*

Specifies the distance in microns that the tap pattern should be shifted to the right.

If the offset distance is a multiple of the tap distance, the effective offset is 0. For example, if the tap distance is 10 microns and the offset is 32 microns, the effective offset is 2 (32 - 3x10).

The default is 0.

-pattern every_row | every_other_row | stagger

Specifies the tap cell insertion pattern. The supported patterns are as follows:

- **every_row** (the default)
  The command adds tap cells to every row using the specified tap distance. The tap distance specified for this pattern should be approximately twice the design rule distance limit.
- **every_other_row**
  The command adds tap cells to every other row for odd rows only. This reduces the number of required tap cells by approximately half as compared to the normal pattern. The tap distance specified for this pattern should be approximately twice the design rule distance limit.
- **stagger**
  The command adds tap cells to every row. The tap cells on even rows are offset with half the tap distance relative to the odd rows, producing a checkerboard-like pattern. This pattern also reduces the number of required tap cells by approximately half as compared to the normal pattern. The tap distance specified for this pattern should be approximately four times the design rule distance limit.

-insert_into_blocks

Inserts tap cells inside the child blocks.

-skip_fixed_cells

Prevents the command from inserting tap cells in locations that are occupied by fixed cells.

When you use this option, the tool treats a fixed cell like a blockage, which breaks the row and might cause a tap cell to be inserted on each side of the fixed cell. To prevent the insertion of tap cells that overlap fixed cells without breaking the row, use the **-preserve_distance_continuity** option with the **-skip_fixed_cells** option.

By default, the command inserts a tap cell in each calculated tap location, even if it is occupied by a fixed cell.

-prefix *cell_prefix*

Specifies the prefix for the created tap cells.

When you use this option, the command uses the following naming convention for the inserted tap cells:

tapfiller!*cell_prefix*!*lib_cell*!*number*

By default, no prefix is added and the tool uses the following naming convention for the inserted tap cells:

tapfiller!*lib_cell*!*number*

-separator *cell_separator*

Specifies the separator character that is used when composing the instance name of the tap cell.

The default is "!".

-at_distance_only

Creates tap cells only at the specified distance, distance/2, or distance/4 (stagger mode). Note that using this option can cause DRC violations.

-enable_prerouted_net_check

When this option is turned on prerouted net checking is performed for tap-boundary cells. Prerouted net checking is not performed on other boundary cells whether or not this option is specified.

-min_horizontal_periphery_spacing *number_of_unit_tiles*

Specifies the minimum horizontal spacing measured in unit tiles from a tap cell that is directly above or below a blocked area to the beginning or end of that placeable row. This option requires the **-skip_fixed_cells** option.

-no_abutment

When this option is specified vertical abutment is enforced during tap cell creation. This means that tap cells are created in a manner where the top or bottom of the tap cells do not overlap with other tap cells or other cells specified by the **-no_abutment_cells** option. Additionally, when this option is specified no abutment horizontal spacing and corner spacing rules are also enforced with the parameters specified to the **-no_abutment_horizontal** and **-no_abutment_corner_spacing** options respectively.

-no_abutment_horizontal_spacing *number_of_unit_tiles*

This options requires the **-no_abutment** option and specifies the minimum horizontal spacing between two different tap cells that are on the same row or a tap cell and another cell specified by the **-no_abutment_cells** option. The default value for this option is 1, which means there must be at least a distance of one unit tile measured horizontally between the cells.

-no_abutment_corner_spacing *number_of_unit_tiles*

This options requires the **-no_abutment** option and specifies the minimum horizontal corner spacing between two different tap cells that are on adjacent rows row or a tap cell and another cell specified by the **-no_abutment_cells** option. The default value for this option is 0, which means that the corners of the cells may touch. For example, if this option is set to 2, it means that there must be at least a distance of two unit tile measured horizontally between the corners of cells in vertically adjacent rows.

-no_abutment_cells *lib_cells*

This options requires the **-no_abutment** option. When this option is specified, tap cell are created so that instead of just complying with no abutment rules with other tap cells, the created tap cells also comply with no abutment rules with other cells that use the library cell passed to this option. This option applies to all three no abutment rules, the no vertical abutment, no abutment horizontal spacing and the no abutment corner spacing rules.

-row_end_tap_bypass

Specifies that the boundary cells have taps.

Note that the command does not check if the boundary cells contain taps nor does it check if the boundary cells exist. It assumes that taps are available at both ends of each row and inserts the tap cells accordingly.

-preserve_distance_continuity *lib_cells*

Ensures that the tap distance is maintained even when a tap cell cannot be placed in a certain location due to fixed instances of the specified library cells.

When you use this option, the command checks if the calculated tap location is occupied by a fixed instance of the specified library cells. If so, the command shifts the tap cell to the left of the fixed cell instance, if possible. Otherwise, it shifts the tap cell to the right of the fixed cell instance. Only the affected tap cell is shifted; the next tap cell remains at its original tap location. To prevent DRC violations, you might need to specify a smaller tap distance when you use this option.

This option affects tap cell insertion only when used with the **-skip_fixed_cells** option.

-multiple_tap_cells

Specifies the tap cells to be used for insertion and the properties of those tap cells, including latch-up coverage distance, the minimum required to the edge of the row, type of left side coverage and right side coverage. The syntax is follows:

-multiple_tap_cells { {tap1 tapCoverageDist1 minRowEdgeDist1 leftCoverage1 rightCoverage1} {tap2 tapCoverageDist2 minRowEdgeDist2 leftCoverage2 rightCoverage2} }

The left side coverage option specifies whether the coverage is provided for the left nwell and pwell half of the cell and can be any of one the following strings: left_full, left_nwell_only, left_pwell_only, left_none.

The right side coverage option specifies whether the coverage is provided for the right nwell and pwell half of the cell and can be any of one the following strings: right_full, right_nwell_only, right_pwell_only, right_none.

-min_row_edge_distance_layer_override *layerMinRowPair*

This option takes a set of layer and distance pairs. When this option is specified the minimum row edge distance is set to the distance value for all rows where the beginning or end of the row is covered by the specified layer.This option requires the **-multiple_tap_cells** option.

This option should be used for all processes that have requirements for cells being within a minimum distance of the edge of the row. This option does not require multiple tap cells to be used. The option can be run with only one tap library cell.

-other_cells_with_well_taps *table*

Specifies cells that are not tap cells that provide latch-up protection and the properties of those cells, including latch-up coverage distance, the minimum required to the edge of the row, type of left side coverage and right side coverage. The syntax is the same as for the -multiple_tap_cells option.

-cells_not_breaking_well_continuity *cell_names*

Specifies the library reference cells that do break the continuity of either nwell or the pwell half of the specified cell.

-cells_breaking_nwell_continuity *cell_names*

Specifies the library reference cells that break the continuity of the half of the cell that is covered by the nwell. Cells specified by this option allow tap cell coverage to propagate through the half of the cell that is covered by the pwell but not the nwell.

-cells_breaking_pwell_continuity *cell_names*

Specifies the library reference cells that break the continuity of the half of the cell that is covered by the pwell. Cells specified by this option allow tap cell coverage to propagate through the half of the cell that is covered by the nwell but not the pwell.

DESCRIPTION

This command adds tap cells to the design. A tap cell is a special cell with a well tie, substrate tie, or both. Tap cells are typically used when most or all standard cell in the library contain no substrate or well taps.

The design rules typically specify a distance limit from every transistor of a standard cell to a well or substrate tie. This command is used to create tap cells before global placement so that all standard cells that are subsequently placed can satisfy the distance limit because of the created tap cells.

You should specify the tap distance and offset based on your specific design rule's distance limit. The command has no knowledge of the design rule's distance limit. A visual check is recommended after running this command to ensure that all standard cell placement areas are properly protected by tap cells.

You can select several tap cell placement patterns. In the following situations, the tap cells are shifted from their calculated tap locations:

- If the **place.legalize.enable_pin_color_alignment_check** application option is **true** (its default is **false**) and the calculated tap location would violate the double-patterning spacing rules.
- If the **-skip_fixed_cells** and **-preserve_distance_continuity** options are used and the tap location is occupied by a fixed instance of the specified library cells.

In these situations, you should reduce the tap distance slightly to avoid signoff DRC errors due to the shifting.

EXAMPLES

The following example creates tap cells. The tap cell array will contain instances of the Cell1 cell in every row. The distance between tap cells is 30 microns.

```
prompt> create_tap_cells -lib_cell myLib/Cell1 \
  -distance 30 -pattern every_row
```

SEE ALSO

create_boundary_cells(2)
create_stdcell_fillers(2)
place.legalize.enable_pin_color_alignment_check(3)

create_tap_meshes

Creates a tap mesh in the specified region.

SYNTAX

status **create_tap_meshes**
-lib_cell *cell_name*
-bbox *coordinates*"
-mesh_window *array_spec*
[-prefix *cell_prefix*]
[-orientation *cell_orientation*]

Data Types

cell_name string
coordinates rectangle
array_spec list
cell_prefix string
cell_orientation string

ARGUMENTS

-lib_cell *cell_name*

Specifies the library reference cell to be used as a tap cell. You must specify a single library cell. This is a required option.

-bbox *coordinates*

Specifies the rectangular tap insertion region in microns. Specify the rectangle as a list of four coordinates, {{llx lly} {urx ury}}, which represents the lower-left and upper-right corners of the rectangle.

This is a required option.

-mesh_window *array_spec*

Specifies x-pitch and y-pitch for the tap cell insertion. This is a required option.

-prefix *cell_prefix*

Specifies the prefix for the created tap cells.

When you use this option, the command uses the following naming convention for the inserted tap cells:

<*cell_prefix*>__<*lib_cell*>_R#_C#_*number*

By default, no prefix is added and the tool uses the following naming convention for the inserted tap cells:

headerfooter__<*lib_cell*>_R#_C#_*number*

-orientation ***cell_orientation***

Specifies the orientation of the placement of the tap cells. The valid values are: R0, R90, R180, R270, MX, MXR90, MY, MYR90.

The default is R0.

DESCRIPTION

This command creates a tap mesh, which is a two-dimensional array of tap cells with a tap cell in each mesh window. A tap mesh is typically inserted outside of a place and route block.

When inserting the tap mesh, the command does not honor the standard cell rows or sites. It places a tap cell in the lower-left corner of each mesh window. If that location is blocked, the command uses the closest available location, which is measured as the Manhattan distance to the lower-left corner of the mesh window. If there is no available location, the command issues a warning and does not place a tap cell in that mesh window.

EXAMPLES

The following example places tap cells in a mesh window that has an x-pitch of 10 and a y-pitch of 20.

```
prompt> create_tap_meshes -lib_cell myLib/Cell1 \
  -bbox {{0 0} {100 100}} -mesh_window {10 20}
```

SEE ALSO

create_dense_tap_cells(2)
create_exterior_tap_walls(2)
create_interior_tap_walls(2)
create_tap_cells(2)

create_taps

Creates voltage source objects (taps) to be used in the rail analysis.

SYNTAX

status **create_taps**
[-top_pg]
[-of_objects *object_list*]
[-import *file_name*]
[-point {*X-coord Y-coord*}]
[-layer *number_or_name*]
[-supply_net *supply_nets*]
[-nocheck]
[-snap_distance *snap_distance*]
[-region {{*left top*} {*right bottom*}}]
[-xpitch *X-pitch*]
[-ypitch *Y-pitch*]
[-xoffset *X-offset*]
[-yoffset *Y-offset*]
[-direction all | v | h]

Data Types

object_list list
file_name string
X-coord real
Y-coord real
number_or_name string
file_name string
supply_nets list
snap_distance real

ARGUMENTS

-top_pg

Indicates that tap points are to be created for all top-level supply (power or ground) pins of the design. This option cannot be used with the **-of_objects**, **-import**, **-point**, or **-snap_distance** option. The **-layer**, **-supply_net**, and **-nocheck** options are ignored if the **-top_pg** option is specified.

-of_objects *object_list*

Creates tap points on the specified object which can be either cell or lib_cell. By default, taps are created on all supply power and ground pins of the object from *object_list*. The **-supply_net** option can be used to determine only which supply power or ground

pins can be put on the taps. If the **-point**, **-layer**, and **-supply_net** options are specified at the same time, one tap is created at the specified physical location; the power of the tap is specified by the **-supply_net** option, relative to the origin of each object. Object names are used as tap names, in the format net_name:cell_name:pin_name.

This option cannot be used with the **-top_pg**, **-import**, or **-snap_distance** option. The **-nocheck** option is ignored if the **-of_objects** option is specified.

-import ***file_name***

Reads in the information from a tap file that defines tap XY locations, layer numbers and supply nets, and so on.

The supported tap file format is as follows:

<net_name> <layer_number> <X-coord> <Y-coord> [*R(<R_in_Ohm>)*] [*L(<L_in_Henry>)*] [*C(<C_in_Farad>)*]

This option cannot be used with the **-top_pg**, **-of_objects**, **-point**, **-layer**, or **-supply_net** option.

-point {*X-coord Y-coord*}

Defines the location where the tap is to be created, in terms of the *X-coord* and *Y-coord* location on the XY plane. When the **-of_objects** option is specified also, then XY location is relative to the origin of the specified objects. Otherwise, XY location is relative to the origin of the current top-level design. Moreover, this option must be accompanied by the **-layer** and **-supply_net** options to define a physical location associated with a given supply net in the three-dimensional space at which the tap is to be created. This option cannot be used with the **-top_pg** or **-import** option. If a tap point is defined in a location where another tap already exists, the tool issues a warning message and replaces the old tap with new one.

-layer ***number_or_name***

Defines the layer at which the tap point is to be created. The layer can be specified as either a layer number or a layer name. This option must be accompanied by the **-point** and **-supply_net** options to define a physical location associated with a given supply net in the three-dimensional space at which the tap is to be created. In addition, this option cannot be used with the **-import** option.

-supply_net ***supply_nets***

Specifies the supply net with which this tap point is associated. If both the **-point** and **-layer** options are used, this option is required and only one supply net should be specified. If there are more than one supply net, then only the first supply net is used with the tool warning message. If the **-of_objects** option is used, then a collection of supply nets is acceptable. This option cannot be used with the **-import** option.

-nocheck

Specifies that tap points are not checked to see if they touch any cross-hierarchy (including soft or hard macros) geometry. This option cannot be used with the **-snap_distance** option. In addition, this option is ignored when you specify the **-top_pg** or **-of_objects** option.

-snap_distance ***snap_distance***

Defines the snap distance in microns. When a given tap location does not touch a layout geometry of a supply net, the tool searches for legal locations within the specified distance by expanding to the left, right, bottom and top corners. This option cannot be used with the **-top_pg**, **-of_objects**, or **-nocheck** option.

-region {{*left top*} {*right bottom*}}

Defines a region to place pitched taps. This option requires both the **-xpitch** and **-ypitch** options and cannot be used with the **-snap_distance** option.

-xpitch ***X-pitch***

Defines a fixed interval along the X-axis to place pitched taps. This option requires both the **-xpitch** and **-ypitch** options and cannot be used with the **-snap_distance** option.

-ypitch ***Y-pitch***

Defines a fixed interval along the Y-axis to place pitched taps. This option requires both the **-xpitch** and **-ypitch** options and cannot be used with the **-snap_distance** option.

-xoffset ***X-offset***

Defines an offset on the X-axis to begin placing pitched taps. This option requires both the **-xpitch** and **-ypitch** options and cannot be used with the **-snap_distance** option.

-yoffset ***Y-offset***

Defines an offset on the the Y-axis to begin placing pitched taps. This option requires both the **-xpitch** and **-ypitch** options and cannot be used with the **-snap_distance** option.

-direction ***all*** | ***v*** | ***h***

Defines a wire direction for which shapes pitched taps can be placed upon. This option requires both the **-xpitch** and **-ypitch** options and cannot be used with the **-snap_distance** option.

DESCRIPTION

The **create_taps** command creates tap objects which are considered as part of supply network during rail analysis.

Taps are used to model the external voltage source environment in which the device under analysis (DUA) operates. They are not part of the design itself, but can be thought of as virtual models of voltage sources. For some analysis types, taps are treated only as sources of ideal voltage. For other analysis types, taps represent the package in which the DUA is contained. In these cases the electrical behavior can be significantly more complex than a simple ideal voltage source.

When a physical location for a tap object is specified via the **-point** and **-layer** options, the tap is virtually connected to the design at that location. If there is no supply (or supply pin) net or via shape at the specified location, which means there is no conductive path to the supply network, the tap has no rail effect.

By default, cross-hierarchy (including soft and hard macros) geometry-touch checking is always performed during tap creation. The checking result is updated to the *is_valid_location* attribute of the tap object from an initial unknown status. Note that the tool drops taps with invalid locations. To disable this checking, use the **-nocheck** option.

If a tap already exists at a location when the **create_taps** command is executed, the new tap will be dropped with a warning message.

Two or more tap points might be created close to one another. This can occur when you run the **create_taps** command multiple times to specify taps in different ways. If taps are placed too close to one another, the effective resistance between them might be too small to have an impact on the rail result. This means these taps effectively act as one single tap point. For example, if the taps are sufficiently separated at opposite ends of a relatively long segment of a net shape, there is non-trivial resistance between them, which alters the rail results more substantially.

When a *lib_cell* is provided within the *object_list* parameter, the tap points are created on all instances of that *lib_cell* used in the design.

If you have an ASCII file that describes tap information, use the **-import** option to read in this file and create taps accordingly. The format of the file is shown in the **-import** option description.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example imports taps from a tap file.

```
prompt> sh cat tap_file

VDD 31 269.900 409.740
VSS 31 218.500 409.740

prompt> create_taps -import tap_file
prompt> report_taps

Name     Net  Point          Layer  Is_valid
-------------------------------------------------
Tap_24   VDD  269.900 409.740  31     true
Tap_25   VSS  218.500 409.740  31     true
```

The following example creates two taps by specifying their layers, locations, and associated supply nets.

```
prompt> create_taps -point {269.900 409.740} -layer 31 -supply_net VDD
prompt> create_taps -point {319.500 409.740} -layer METAL4 \
  -supply_net VSS
```

The following example creates taps at all top-level supply pins.

```
prompt> create_taps -top_pg
```

The following example create taps by objects which could be cells or lib_cells.

```
prompt> create_taps -of_objects [get_cell U5]
prompt> create_taps -of_objects [get_cell U5] -supply_net VDD
prompt> create_taps -of_objects [get_cell U5] -point {2.8000 1.8000} \
  -layer 31 -supply_net VDD
prompt> create_taps -of_objects [get_lib_cell */EDF]
prompt> create_taps -of_objects [get_lib_cell */EDF] -supply_net VDD
prompt> create_taps -of_objects [get_lib_cell */EDF] \
  -point {2.8000 1.8000} -layer 31 -supply_net VDD
```

The following example creates pitched taps within the region {{23.285 10.63} {23.285 11.39}} spaced by an interval of 0.38 length units along the X-axis and Y-axis and touching horizontal net shapes of the supply net VDD and the layer 32. Since an offset is not specified, the pitched taps have their intervals begin at the origin.

```
prompt> create_taps -region {{23.285 10.63} {23.285 11.39}} \
  -xpitch 0.38 -ypitch 0.38 -supply_net VDD -layer 32 \
  -direction h
```

SEE ALSO

get_taps(2)
remove_taps(2)
report_taps(2)

create_targeted_boundary_cells

Creates and places boundary cells around target objects in current design.

SYNTAX

status **create_targeted_boundary_cells**
-target_objects {*target_objects*}
[-include_touching_placement_blockages]
[-ignore_row_orientation]
[-prefix *boundary_cell_prefix*]
[-separator *boundary_cell_separator*]
[-left_boundary_cell *lib_cell_name*]
[-right_boundary_cell *lib_cell_name*]
[-bottom_boundary_cells *lib_cell_name*]
[-top_boundary_cells *lib_cell_name*]
[-bottom_left_outside_corner_cell *lib_cell_name*]
[-bottom_right_outside_corner_cell *lib_cell_name*]
[-top_left_outside_corner_cell *lib_cell_name*]
[-top_right_outside_corner_cell *lib_cell_name*]
[-bottom_left_inside_corner_cells *lib_cell_name*]
[-bottom_right_inside_corner_cells *lib_cell_name*]
[-top_left_inside_corner_cells *lib_cell_name*]
[-top_right_inside_corner_cells *lib_cell_name*]
[-mirror_left_outside_corner_cell]
[-mirror_right_outside_corner_cell]
[-mirror_left_inside_corner_cell]
[-mirror_right_inside_corner_cell]
[-mirror_left_boundary_cell]
[-mirror_right_boundary_cell]
[-mirror_left_inside_horizontal_abutment_cell]
[-mirror_right_inside_horizontal_abutment_cell]
[-tap_distance *distance*]
[-top_tap_cell *lib_cell_name*]
[-bottom_tap_cell *lib_cell_name*]

Data Types

target_objects list
boundary_cell_prefix string
boundary_cell_separator string
lib_cell_name string
orientation orientation in DEF syntax
distance float

ARGUMENTS

-target_objects {*target_objects*}

Specifies the target objects to be around with boundary cells in current design. Objects including macro, voltage area, voltage area shape, placement blockage, route blockage and core area can be specified. This option is a required option.

-include_touching_placement_blockages

Specifies that when boundary cells are created and placed, placement blockages those touch specified target objects should be included. By default, no touching placement blockages are included.

-ignore_row_orientation

Specifies that when flipping between top and bottom.

-prefix *boundary_cell_prefix*

Specifies the prefix for the created boundary cells. By default, no prefix is added.

-separator *boundary_cell_separator*

Specifies the separator character that is used when composing the instance name of the boundary cell. The boundary cell instance name consists of a prefix, the library reference cell name and an incrementing number. For example, the instance name "boundarycell!MY_CELL!25" has the boundary cell prefix, the MY_CELL library reference cell name, the incrementing number 25, and the boundary cell separator !. This naming convention allows you to do pattern matching selection of the boundary cells. The default is "!".

-left_boundary_cell *lib_cell_name*

Specifies the library cell to be placed at the beginning of each cell row.

-right_boundary_cell *lib_cell_name*

Specifies the library cell to be placed at the end of each cell row.

-bottom_boundary_cells *lib_cell_name*

Specifies the library cells to be placed along the bottom object boundaries. In a double-back design, adjacent rows are flipped; the bottom boundary cell is used on unflipped bottom boundary rows and on flipped top boundary rows.

-top_boundary_cells *lib_cell_name*

Specifies the library cells to be placed along the top of object boundaries. In a double-back design, adjacent rows are flipped; the top boundary cell is used on unflipped top boundary rows and on flipped bottom boundary rows.

-bottom_left_outside_corner_cell *lib_cell_name*

Specifies the library cell to be placed at bottom-left outside corners. An outside corner is a corner with a 90-degree inside angle. A bottom corner is a corner that is on a bottom boundary. In a double-back design, adjacent rows are flipped; the bottom corner cell is used on unflipped bottom boundary rows and on flipped top boundary rows.

-bottom_right_outside_corner_cell *lib_cell_name*

Specifies the library cell to be placed at bottom-right outside corners. An outside corner is a corner with a 90-degree inside angle. A bottom corner is a corner that is on a bottom boundary. In a double-back design, adjacent rows are flipped; the bottom corner cell is used on unflipped bottom boundary rows and on flipped top boundary rows.

-top_left_outside_corner_cell *lib_cell_name*

Specifies the library cell to be placed at top-left outside corners. An outside corner is a corner with a 90-degree inside angle. A top corner is a corner that is on a top boundary. In a double-back design, adjacent rows are flipped; the top corner cell is used on unflipped top boundary rows and on flipped bottom boundary rows.

-top_right_outside_corner_cell *lib_cell_name*

Specifies the library cell to be placed at top-right outside corners. An outside corner is a corner with a 90-degree inside angle. A top corner is a corner that is on a top boundary. In a double-back design, adjacent rows are flipped; the top corner cell is used on unflipped top boundary rows and on flipped bottom boundary rows.

-bottom_left_inside_corner_cells *lib_cell_name*

Specifies the library cell to be placed at the bottom-left of inside corners. An inside corner is a corner with a 270-degree inside angle. A left inside corner cell is an inside corner cell on the left end of the horizontal edge that makes up the inside corner.

-bottom_right_inside_corner_cells *lib_cell_name*

Specifies the library cell to be placed at the bottom-right of inside corners. An inside corner is a corner with a 270-degree inside angle. A right inside corner cell is an inside corner cell on the right end of the horizontal edge that makes up the inside corner.

-top_left_inside_corner_cells *lib_cell_name*

Specifies the library cell to be placed at the top-left of inside corners. An inside corner is a corner with a 270-degree inside angle. A left inside corner cell is an inside corner cell on the left end of the horizontal edge that makes up the inside corner.

-top_right_inside_corner_cells *lib_cell_name*

Specifies the library cell to be placed at the top-right of inside corners. An inside corner is a corner with a 270-degree inside angle. A right inside corner cell is an inside corner cell on the right end of the horizontal edge that makes up the inside corner.

-mirror_left_outside_corner_cell

Places left outside corner cells in a mirrored orientation.

-mirror_right_outside_corner_cell

Places right outside corner cells in a mirrored orientation.

-mirror_left_inside_corner_cell

Places left inside corner cells in a mirrored orientation.

-mirror_right_inside_corner_cell

Places right inside corner cells in a mirrored orientation.

-mirror_left_boundary_cell

Places left boundary cells in a mirrored orientation.

-mirror_right_boundary_cell

Places right boundary cells in a mirrored orientation.

-mirror_left_inside_horizontal_abutment_cell

Places left inside horizontal abutment cells in a mirrored orientation.

-mirror_right_inside_horizontal_abutment_cell

Places right inside horizontal abutment cells in a mirrored orientation.

-tap_distance *distance*

Specifies the distance in microns between tap cells.

-top_tap_cell *lib_cell_name*

Specifies the tap cell to be placed at the top boundary. The tool inserts this tap cell at the specified interval on top boundary rows. This tap cell is for the boundary cells and should not be confused with tap cells for standard cells. In a double-back design, adjacent rows are flipped; the top tap cell is used on unflipped top boundary rows and on flipped bottom boundary rows. This option must be used with the **-tap_distance** option.

-bottom_tap_cell *lib_cell_name*

Specifies the tap cell to be placed at the bottom boundary. The tool inserts this tap cell at the specified interval on bottom boundary rows. This tap cell is for the boundary cells and should not be confused with tap cells for standard cells. In a double-back design, adjacent rows are flipped; the bottom tap cell is used on unflipped bottom boundary rows and on flipped top boundary rows. This option must be used with the **-tap_distance** option.

DESCRIPTION

This command adds boundary cells around the boundaries of target objects, such as voltage areas, macros, blockages, and the core area.

EXAMPLES

The following example creates boundary cells around target placement blockages on both the left and the right edges of boundaries.

```
prompt> create_targeted_boundary_cells -target_objects {PB_7 PB_8} \
        -left_boundary_cell myLib/CellLeft -right_boundary_cell myLib/CellRight
```

SEE ALSO

set_boundary_cell_rules(2)
report_boundary_cell_rules(2)
remove_boundary_cell_rules(2)
compile_boundary_cells(2)
check_boundary_cells(2)
compile_targeted_boundary_cells(2)
check_targeted_boundary_cells(2)

create_tech

Creates a technology in the given library

SYNTAX

collection **create_tech**
[-library]
[-force]
name

Data Types

name string

ARGUMENTS

-library

Technology will be created in this library. If not specified then current library will be used.

-force

By default if a library already has a tech then create_tech will return error. But with this option, create_tech will replace the existing tech with new one.

name

Name of the technology.

DESCRIPTION

create_tech command creates a "tech object", the object to contain all technology related data. Typically the tech object is created from a tech file. This command provides a user interface to create the tech object directly from the Tcl interface.

The tech object can be created for a specific library, that can be sepecified as an option to this command. If the library is not specified with this command, the current library is used for tech object creation. If there is no current library set and no library specified with this command, the command will return an error.

create_tech fails if the tech already exists for the given library, however tech object creation can be forced by using -force option.

If the command is successful, it returns the newly-created tech as a single-element collection. If the command fails, it returns TCL_ERROR with appropriate error message. If there is a syntax error, it returns a TCL_ERROR.

EXAMPLE

The following example creates a technology object in the given library

```
prompt> create_tech -library r4000 tcbn90g
{r4000}
prompt> get_techs -of_objects [get_libs r4000]
{tcbn90g}
```

The following example replaces the existing technology object by creating a new technology object in the current library.

```
prompt> create_tech -force tcbn90g
{tcbn90g}
prompt> get_techs -of_objects [current_lib]
{tcbn90g}
```

SEE ALSO

remove_tech(2)
get_techs(2)

create_terminal

Creates a terminal on a port in the current design.

SYNTAX

collection **create_terminal**
-port *port*
-of_objects shapes_and_via_shapes
[-object *shape_or_via*]
[-direction *access_dir*]
[-eeq_class *eeq_class_value*]
[-name *terminal_name*]

Data Types

port collection
shapes_and_via_shapes collection
shape_or_via collection
access_dir list
eeq_class_value integer
terminal_name string

ARGUMENTS

-port *port*

Specifies the port on which to create the terminal. The option "-port" and "-object" should be specified together.

-object *shape_or_via*

Specifies the shape or via assigned to the terminal. This shape or via is owned by the terminal and defines its geometry. It is an error to include a shape or via which has already been assigned to another terminal. Text shapes may not be assigned to a terminal. The option "-port" and "-object" should be specified together.

-of_objects *shapes_and_via_shapes*

Specifies the net shapes and vias which are to be duplicated as terminal shapes. These copied net shapes and via geometries are owned by the terminals and defines their geometry. The shapes and the via geometries should have a valid net id assigned, otherwise they will be ignored. Text shapes may not be assigned to a terminal. This option is mutually exclusive with option "-port". The options "-eeq_class" and "-name" are ignored when used with "-of_objects" option.

-direction *access_dir*

Specifies the list of terminal access directions. Possible values are none, all, or one or more of: left, right, bottom, top. Default value is all.

-eeq_class *eeq_class_value*

Specify value for eeq_class attribute for the terminal. Value range is 0-65535.

-name *terminal_name*

Specifies the name of the terminal. If not specified, the name will be automatically generated.

DESCRIPTION

This command creates a new terminal and returns a collection containing the new terminal. Each shape or via may belong to one and only one terminal. A single rect, polygon, path, simple via, simple array via, or custom via may belong to the terminal. A text shape may not belong any terminal.

EXAMPLES

The following example creates a terminal with a shape.

```
prompt> create_terminal -port port1 -object [get_shapes PATH_16_40] -direction "all"
```

The following example creates a terminal with a via.

```
prompt> create_terminal -port port1 -object [get_vias VIA_S_1] -direction "all"
```

The following example creates terminals from net shapes and via.

```
prompt> create_terminal -of_objects {PATH_28_1393 PATH_18_2008  VIA_S_5491}
{TM_129 TM_130 TM_131}
```

SEE ALSO

get_terminals(2)
remove_terminals(2)
get_shapes(2)
get_vias(2)

create_terminals_for_pins

Create terminals for pins based on user constraints.

SYNTAX

status **create_terminals_for_pins**
[-pins *pins*]
[-nets *nets*]
[-ports *ports*]
[-bundles *bundles*]

Data Types

pins	collection
nets	collection
ports	collection
bundles	collection

ARGUMENTS

-pins *pins*

Specifies a list of pins for creating terminals.

-nets *nets*

Specifies a list of nets for creating terminals.

-ports *ports*

Specifies a list of ports for creating terminals.

-bundles *bundles*

Specifies a list of bundles for creating terminals.

DESCRIPTION

This command does quick pin assignment based on user constraints without doing global route and no feedthrough creation. For the pins to be created, they at least should have a side constraint. If possible, user should also consider providing offset or range constraints, in conjunction with layer constraints. During the pin creation, the command starts from the center of the constrained

region and search for the leagl pin location. The finer the constraints are, the fast the pin creation runtime will be.

The command honors user provided individual, bundle, block pin constraints, NDR rules, and recognize PG mesh and pin blockages. It also honors the spacing rules and size rules defined in technology files that the command **place_pins** honors. If multiple pin constraints are provide but are conflict to each other, the prioritiy from the highest to the lowest is NDR, individual pin constraints, bundle pin constraints, block pin constraints. If the user provided constraints conflict with technology rules, the values defined by the technology rules are honored.

The command does not consider alignment or abutment during pin creation. Its main purpose is to create physical pins as closest as possible to the constrained region, i.e. the shortest distance to the constrained region is the objective function. The intended usage of this command is to quickly create a collection of pins, interactively update them from GUI if desired, then fix them. These fixed pins will act as anchor points for later global route based pin creation to create feedthroughs. It is not intended to replace the command **place_pins**, nor a different flavor of **place_pins -nets_to_exclude_from_routing**.

The command will not re-create pins for fixed pins.

EXAMPLES

The following example creates terminals for pins

```
prompt> create_terminals_for_pins -pins [get_pins ]
1
```

SEE ALSO

place_pins(2)

create_test_protocol

Creates a test protocol based on user specifications.

SYNTAX

<return_value> **create_test_protocol**

ARGUMENTS

The create_test_protocol command has no arguments.

DESCRIPTION

The create_test_protocol command creates a test protocol for the cur- rent design based on user specifications issued prior to running this command. The specifications are made using commands such as set_dft_signal.

This command removes any protocol that is present in memory due to a previous execution of create_test_protocol. However, if the protocol is present in memory due to a previous execution of read_test_protocol, it issues a warning and does not create a new test protocol.

This command checks whether the user-specified values are consistent with each other. If they are not, it issues an error and does not generate a protocol.

EXAMPLES

The following example creates a test protocol in memory for the current design:

```
prompt> create_test_protocol
```

SEE ALSO

set_dft_signal(2)
insert_dft(2)

create_topological_constraint

Creates a new topological pin feedthrough constraint.

SYNTAX

create_topological_constraint
[-start_sides *side_numbers*]
[-start_offset *offset*]
[-start_offset_range *offset_range*]
[-start_layers *layers*]
[-end_sides *side_numbers*]
[-end_offset *offset*]
[-end_offset_range *offset_range*]
[-end_layers *layers*]
-start_object *object*
-end_object *object*
owner

Data Types

side_numbers list
offset float
offset_range float
layers collection
object collection
owner collection

ARGUMENTS

-start_sides *side_numbers*

Specifies the allowed side numbers on the start object for the pin-connection to go through. For a rectangular or rectilinear shape, the leftmost edge is the side number 1. If there are multiple leftmost edges, then edge 1 is the lowest leftmost edge. Numbering continues from the leftmost edge in the clockwise direction. This option is applicable only if the start object is of type cell, voltage-area or voltage-area-region. This is an optional option. When not specified the pin may be placed on any side.

-start_offset *offset*

Specifies the offset distance from the edge's starting point. The starting point is the first point on the edge in the clockwise direction. This is an optional option and can be specified only when the *-start_sides* option is specified.

-start_offset_range *offset_range*

Specifies the distance from the offset that a pin may be placed. This is an optional option and can be specified only when the *-start_offset* option is specified. When specified, the pin can be placed anywhere between (offset-range) and (offset+range).

-start_layers *layers*

Specifies the routing layers on which the pins may be created. This is an optional option. When no layers are specified then any layer may be used for pin placement.

-end_sides *side_numbers*

Specifies the allowed side numbers on the end object for the pin-connection to go through. For a rectangular or rectilinear shape, the leftmost edge is the side number 1. If there are multiple leftmost edges, then edge 1 is the lowest leftmost edge. Numbering continues from the leftmost edge in the clockwise direction. This option is applicable only if the start object is of type cell, voltage-area or voltage-area-region. This is an optional option. When not specified the pin may be placed on any side.

-end_offset *offset*

Specifies the offset distance from the edge's starting point. The starting point is the first point on the edge in the clockwise direction. This is an optional option and can be specified only when the *-end_sides* option is specified.

-end_offset_range *offset_range*

Specifies the distance from the offset that a pin may be placed. This is an optional option and can be specified only when the *-end_offset* option is specified. When specified, the pin can be placed anywhere between (offset-range) and (offset+range).

-end_layers *layers*

Specifies the routing layers on which the pins may be created. This is an optional option. When no layers are specified then any layer may be used for pin placement.

-start_object *object*

Specifies the object that forms the start of the connection. Valid object include an object of a cell, a voltage-area, a voltage-area-region, a port or a pin. Unless the object is of type cell, the values in the *-start_object* and *-end_object* should be different. If the object is of type port or pin, then it must be logically connected to the owner object (net or bundle).

-end_object *object*

Specifies the object that forms the end of the connection. Valid object include an object of a cell, a voltage-area, a voltage-area-region, a port or a pin. Unless the object is of type cell, the values in the *-end_object* and *-start_object* should be different. If the object is of type port or pin, then it must be logically connected to the owner object (net or bundle).

owner

Specifies the objects that will own a copy of the newly created constraint. Valid objects include a homogenous or heterogeneous collection of nets or bundles.

DESCRIPTION

The **create_topological_constraint** command creates topological pin feedthrough constraint with the specified parameters in the current block.

The topological constraints owned by a net or a bundle should be unique. If a topological constraint similar to the one being created is already owned by the net or the bundle, then the command errors out reporting the same.

EXAMPLES

The following example creates a topological pin feedthrough constraint with the cell1 and cell2 cells as the start and end object respectively. The topological constraint are owned by the net named net1.

```
prompt> create_topological_constraint net1 \
  -start_object cell1 -end_object cell2
```

The following example creates a topological pin feedthrough constraint with port "port1" and pin "pin1" as start and end object respectively. The topological constraint will be owned by the bundle bundle1. The topological constraint also specifies the start and end routing layers:

```
prompt> create_topological_constraint bundle1 \
  -start_object port1 -end_object pin1 \
  -start_layers {METAL1} -end_layers {METAL2 METAL3}
```

The following example creates a topological pin feedthrough constraint with cell "cell1" and voltage-area-shape VOLTAGE_AREA_SHAPE_0, owned by the net named net1. The topological constraint also specifies the side numbers, the offset, and the offset range to be used:

```
prompt> create_topological_constraint net1 \
  -start_object cell1 -end_object VOLTAGE_AREA_SHAPE_0 \
  -start_sides {1 2} -end_sides {1 2 3 4} \
  -start_offset 1.2 -end_offset 2.3 \
  -start_offset_range 0.5 -end_offset_range 0.5
```

SEE ALSO

get_topological_constraints(2)
remove_topological_constraints(2)
report_topological_constraints(2)

create_topology_edge

Creates a topology edge.

SYNTAX

collection **create_topology_edge**
[-name *topology_edge_name*]
[-plan *topology_plan*]
[-nodes *node_list*]
[-objects *object_list*]
[-net_estimation_rule *rule_name*]
[-shape_layers *layer_length_width_list*]
[-allow_feedthrough none | select_on | select_off | all]
[-allow_feedthrough_type pure | mixed | any]
[-allow_feedthrough_select *object_list*]
[-allow_flyover none | select_on | select_off | all]
[-allow_flyover_select *object_list*]
[-constraint_groups *constraint_group_list*]
[-group_master *edge_list*]

Data Types

topology_edge_name string
topology_plan collection
node_list collection
object_list collection
rule_name string
layer_length_width_list list of {layer_name length_or_percentage distance} triplets
constraint_group_list collection
edge_list collection

ARGUMENTS

-name *topology_edge_name*

Specifies the name of the topology_edge. The name may not contain the special characters '~' or '/'. If unspecified, the command will generate a name automatically using the prefix "TOPOLOGY_EDGE" with a system-generated number appended.

-plan *topology_plan*

Specifies the topology_plan that will own the topology_edge. If unspecified, the topology_edge will be created in the current topology_plan.

-nodes *node_list*

Specifies a path of topology_nodes on which to create topology_edges, such that if there are N nodes in the list, then (N-1) edges will be created with the specified nodes as endpoints.

-objects *object_list*

Specifies the set of objects to be associated with the topology_edge. The objects may be hierarchical. Allowed object types are pins, ports, nets, supernets, and bundles.

-net_estimation_rule *rule_name*

Specifies the name of the net_estimation_rule of the topology_edge.

-shape_layers *layer_length_width_list*

Specifies the shape_layers of the topology_edge, in the form of a list of {layer_name distance_or_percentage distance} triplets. The second element, "distance_or_percentage", can either be an absolute length for that segment of the edge, or it can be a percentage, in which case it must be appended with the percent symbol. The third element is a distance value representing the width of the edge on that segment.

-allow_feedthrough none | select_on | select_off | all

Specifies the allow_feedthrough setting of the topology_edge.

"none" indicates that no feedthrough is allowed for the edge.

"select_on" indicates that feedthroughs are only allowed through the objects specified in the *-allow_feedthrough_select* option.

"select_off" indicates that feedthroughs are allowed through all objects except those specified in the *-allow_feedthrough_select* option.

"all" indicates that feedthroughs are allowed through all objects.

-allow_feedthrough_type pure | mixed | any

Specifies the allow_feedthrough_type setting of the topology_edge.

-allow_feedthrough_select *object_list*

Specifies the collection of objects for which the *-allow_feedthrough_select* option is applied.

-allow_flyover none | select_on | select_off | all

Specifies the allow_flyover setting of the topology_edge.

"none" indicates that no flyover is allowed for the edge.

"select_on" indicates that flyover is only allowed over the objects specified in the *-allow_flyover_select* option.

"select_off" indicates that flyover is allowed over all objects except those specified in the *-allow_flyover_select* option.

"all" indicates that flyover is allowed over all objects.

-allow_flyover_select *object_list*

Specifies the collection of objects for which the *-allow_flyover_select* option is applied.

-constraint_groups *constraint_group_list*

Specifies the constraint groups to which to add the topology_edge as a member.

-group_master *edge_list*

Specifies the topology_edge group master edge of this topology_edge. Only one topology_edge may be specified, and it must be

in the same topology_plan as this edge.

A topology_edge group is a group of topology_edges sharing the same (or reversed direction) endpoints and coordinate points, as well as selected other attributes. The "master" edge of the group is the edge that governs the attributes of the other edges in the group (the "members").

If this option is specified, then the following options may not be set, as their values will be governed by the master edge: *-allow_feedthrough*, *-allow_feedthrough_type*, *-allow_feedthrough_select*, *-allow_flyover*, *-allow_flyover_select*.

You may set the *-nodes* option if this option is specified, however it must contain the same node endpoints as the master edge, though they may be reversed in direction. If the *-nodes* option is not set, the new edge will have the same endpoints as the master edge.

You may set the *-shape_layers* option if this option is specified, however the value specified may be automatically modified such that the new edge's *points* attribute matches that of the master edge. If the *-shape_layers* option is not set, the new edge will have the same *shape_layers* attribute as the master edge.

DESCRIPTION

The **create_topology_edge** command creates a graph-like edge in a topology_plan. Each topology_edge has two endpoints, *start_node* and *end_node*, each of which may be connected to a topology node, or may be left unconnected.

The name of a topology_edge must be unique within its topology_plan. If the "-plan" option of the command is not specified, then the edge will be created in the "current" topology_plan. Once created, the edge cannot be moved to a different plan, and if the plan is deleted then the edge will also be deleted.

EXAMPLES

The following example creates a topology_edge in the current topology_plan "curplan":

```
prompt> create_topology_edge
{curplan/TOPOLOGY_EDGE0}
```

The following example creates a topology_edge in the topology_plan "myplan", and sets its objects collection to the nets in cell "mycell"'s ref_block:

```
prompt> create_topology_edge -plan myplan -objects [get_nets mycell/*]
{myplan/TOPOLOGY_EDGE0}
```

The following example creates a topology_edge in the current topology_plan "curplan", and that allows feedthrough through all objects except the cells in cell "mycell"'s ref_block:

```
prompt> create_topology_edge -allow_feedthrough select_off -allow_feedthrough_select [get_cells mycell/*]
{myplan/TOPOLOGY_EDGE0}
```

The following example creates 3 topology_edges in the topology_plan "myplan", with the name prefix "myedge", connecting topology_nodes n0, n1, n2, and n3:

```
prompt> create_topology_edge -plan myplan -name myedge -nodes {n0 n1 n2 n3}
{myplan/myedge myplan/myedge_1 myplan/myedge_2}
```

SEE ALSO

create_topology_node(2)
get_topology_nodes(2)
get_topology_edges(2)
remove_topology_nodes(2)
remove_topology_edges(2)
report_topology_plans(2)
current_topology_plan(2)

create_topology_node

Creates a topology node.

SYNTAX

collection **create_topology_node**
[-name *topology_node_name*]
[-plan *topology_plan*]
[-ref_node *topology_node*]
[-cell *cell*]
[-ref_plan *topology_plan*]
[-objects *object_list*]
[-constrained_objects *object_list*]
[-origin *coord*]
[-constraint_type origin | edge | side | bbox | tap]
[-edge *edge_list*]
[-start *dist_or_pct*]
[-end *dist_or_pct*]
[-range *dist_or_pct*]
[-offset *distance*]
[-side *side_list*]
[-alignment left | right]
[-bbox_percentage *float_single_or_pair*]
[-corner_type auto | cross | default | river]
[-constraint_source user | application]
[-halo *distance*]

Data Types

topology_node_name string
topology_plan collection
topology_node collection
cell collection
object_list collection
coord float_pair
edge_list integer_list
dist_or_pct float [+ "%"]
distance float
side_list string

ARGUMENTS

-name *topology_node_name*

Specifies the name of the topology_node. The name may not contain the special character '/'. If unspecified, the command will generate a name automatically using the prefix "TOPOLOGY_NODE" with a system-generated number appended.

-plan *topology_plan*

Specifies the topology_plan that will own the topology_node. If unspecified, the topology_node will be created in the current topology_plan.

-ref_node *topology_node*

Specifies the ref_node of the topology_node. If specified, the topology_node will be in "reference" mode. In reference mode, the topology node will assume the majority of the ref_node's attribute values, and setting those attribute values will result in the values being set on the ref_node.

This option is mutually exclusive with the *-cell* and *-ref_plan* options.

-cell *cell*

Specifies that this topology_node should be in reference mode, and create its ref_node in a new topology_plan in the block referenced by *cell*.

This option is mutually exclusive with the *-ref_node* and *-ref_plan* options.

-ref_plan *topology_plan*

Specifies that this topology_node should be in reference mode, and create its ref_node in *topology_plan*.

This option is mutually exclusive with the *-ref_node* and *-cell* options.

-objects *object_list*

Specifies the set of objects to be associated with the topology_node. The objects may be hierarchical. Allowed object types are pins, ports, cells, voltage_areas, voltage_area_shapes, shapes, placement_blockages, routing_blockages, bounds, and bound_shapes.

-constrained_objects *object_list*

Specifies the set of constrained objects to be associated with the topology_node. The objects may be hierarchical. Allowed object types are pins and ports.

-origin *coord*

Specifies the origin of the topology_node. If unspecified, the origin will be auto-computed based on the other specified options if possible.

-constraint_type origin | edge | side | bbox | tap

Specifies the constraint type of the topology_node.

-edge *edge_list*

Specifies the constraint edge list of the topology_node. Value must be a list of integer values.

-start *dist_or_pct*

Specifies the constraint start of the topology_node. Value may be either a distance or a percentage - to specify a percentage, append a '%' character to the end of the float value.

-end *dist_or_pct*

Specifies the constraint end of the topology_node. Value may be either a distance or a percentage - to specify a percentage, append a '%' character to the end of the float value.

-range *dist_or_pct*

Specifies the constraint range of the topology_node. Value may be either a distance or a percentage - to specify a percentage, append a '%' character to the end of the float value.

-offset *distance*

Specifies the constraint offset of the topolology_node.

-side *side_list*

Specifies the constraint sides of the topolology_node. Value may be a single instance or list of the following: W, N, E, or S.

-alignment left | **right**

Specifies the constraint alignment of the topolology_node.

-bbox_percentage *float_single_or_pair*

Specifies the bbox_percentage of the topology_node. Specify a single value for linear regions, and a float pair for 2D regions.

-corner_type auto | **cross** | **default** | **river**

Specifies the corner_type of the topology_node.

-constraint_source user | **application**

Specifies the constraint_source of the topology_node.

-halo *distance*

Specifies the halo of the topology_node. May only be specified when the *-origin* argument is specified.

DESCRIPTION

The **create_topology_node** command creates a graph-like node or "steiner point" in a topology_plan. topology_nodes serve as endpoints of topology_edges, and have various constraint-related attributes.

The name of a topology_node must be unique within its topology_plan. If the "-plan" option of the command is not specified, then the node will be created in the "current" topology_plan. Once created, the node cannot be moved to a different plan, and if the plan is deleted then the node will also be deleted.

Some topology_nodes may reside in the top level, but reference different nodes, usually at a lower level of hierarchy. A node that references a different node (a "ref_node") is considered to be in "reference mode". When in reference mode, the values of the node's attributes will reflect those of the ref_node. Exceptions are the node's name and the node's edge connectivity-related attributes. When in reference mode, the node's *edges* attribute will report the union of the edges of the node and its ref_node.

EXAMPLES

The following example creates a topology_node in the current topology_plan "curplan":

```
prompt> create_topology_node
{curplan/TOPOLOGY_NODE0}
```

The following example creates a topology_node in the topology_plan "myplan", and sets its objects collection to the cells in cell "mycell"'s ref_block:

```
prompt> create_topology_node -plan myplan -objects [get_cells mycell/*]
{myplan/TOPOLOGY_NODE0}
```

The following example creates a topology_node named "mynode". The node will be in reference mode, and a new topology_node will be created to be its ref_node. The ref_node will be created in a new topology_plan in cell "mycell"'s ref_block:

```
prompt> create_topology_node -name mynode -cell [get_cells mycell]
{curplan/mynode}
```

SEE ALSO

update_topology_node(2)
create_topology_edge(2)
current_topology_plan(2)
get_topology_edges(2)
get_topology_nodes(2)
remove_topology_edges(2)
remove_topology_nodes(2)
report_topology_plans(2)

create_topology_plan

Creates a topology plan in the current block.

SYNTAX

collection **create_topology_plan**
[-name *topology_plan_name*]
[-cell *cell*]
[-objects *object_list*]
[-net_estimation_rule *rule_name*]
[-allow_feedthrough none | select_on | select_off | all]
[-allow_feedthrough_type pure | mixed | any]
[-allow_feedthrough_select *object_list*]
[-allow_flyover none | select_on | select_off | all]
[-allow_flyover_select *object_list*]
[-launch_budget *float*]
[-capture_budget *float*]
[-comment *comment*]

Data Types

topology_plan_name string
cell collection
object_list collection
rule_name string
comment string

ARGUMENTS

-name *topology_plan_name*

Specifies the name of the topology_plan. The name may not contain the special characters '~' or '/'. If unspecified, the command will generate a name automatically using the prefix "TOPOLOGY_PLAN" with a system-generated number appended.

-cell *cell*

Specifies that this topology_plan should be created in the block referenced by *cell*.

-objects *object_list*

Specifies the set of objects to be associated with the topology_plan. The objects may be hierarchical. Allowed object types are pins, ports, nets, supernets, and bundles.

-net_estimation_rule *rule_name*

Specifies the name of the net_estimation_rule of the topology_plan.

-allow_feedthrough none | **select_on** | **select_off** | **all**

Specifies the allow_feedthrough setting of the topology_plan.

"none" indicates that no feedthrough is allowed for the plan.

"select_on" indicates that feedthrough is only allowed through the objects specified in the *-allow_feedthrough_select* option.

"select_off" indicates that feedthrough is allowed through all objects except those specified in the *-allow_feedthrough_select* option.

"all" indicates that feedthrough is allowed through all objects.

-allow_feedthrough_type pure | **mixed** | **any**

Specifies the allow_feedthrough_type setting of the topology_plan.

-allow_feedthrough_select *object_list*

Specifies the collection of objects for which the *-allow_feedthrough_select* option is applied.

-allow_flyover none | **select_on** | **select_off** | **all**

Specifies the allow_flyover setting of the topology_plan.

"none" indicates that no flyover is allowed for the plan.

"select_on" indicates that flyover is only allowed over the objects specified in the *-allow_flyover_select* option.

"select_off" indicates that flyover is allowed over all objects except those specified in the *-allow_flyover_select* option.

"all" indicates that flyover is allowed over all objects.

-allow_flyover_select *object_list*

Specifies the collection of objects for which the *-allow_flyover_select* option is applied.

-launch_budget *float*

Specifies the launch budget of the topology_plan.

-capture_budget *float*

Specifies the capture budget of the topology_plan.

-comment *comment*

Specifies the comment of the topology_plan. This is a general-purpose string that can be used for documentation.

DESCRIPTION

The **create_topology_plan** command creates a new topology_plan in the current block. A topology_plan encapsulates topological planning and constraint data, and owns collections of topology_nodes and topology_edges.

The name of a topology_plan must be unique in the block.

A topology_plan may be set as the "current" topology_plan using the *current_topology_plan* command. Subsequent commands to

create or query topology objects will operate in the context of the current topology_plan.

EXAMPLES

The following example creates a topology_plan in the current block:

```
prompt> create_topology_plan
{TOPOLOGY_PLAN0}
```

The following example creates a topology_plan with name "myplan" and sets its objects collection to the nets in cell "mycell"'s ref_block:

```
prompt> create_topology_plan -name myplan -objects [get_nets mycell/*]
{myplan}
```

The following example creates a topology_plan that allows feedthrough through all objects except the cells in cell "mycell"'s ref_block:

```
prompt> create_topology_plan -allow_feedthrough select_off -allow_feedthrough_select [get_cells mycell/*]
{TOPOLOGY_PLAN0}
```

SEE ALSO

create_topology_edge(2)
create_topology_node(2)
current_topology_plan(2)
get_topology_plans(2)
remove_topology_plans(2)
report_topology_plans(2)

create_topology_repeater

Creates a topology repeater.

SYNTAX

collection **create_topology_repeater**
-edge *topology_edge*
[-name *topology_repeater_name*]
[-type repeater | sequential]
[-offset *distance*]
[-objects *object_list*]

Data Types

topology_edge collection
topology_repeater_name string
distance float

ARGUMENTS

-edge *topology_edge*

Specifies the topology_edge on which to create the topology_repeater.

-name *topology_repeater_name*

Specifies the name of the topology_repeater. The name may not contain the special characters '~' or '/'. If unspecified, the command will generate a name automatically using the prefix "TOPOLOGY_REPEATER" with a system-generated number appended.

-type repeater | sequential

Specifies the type of the topology_repeater.

-offset *distance*

Specifies the offset of the topology_repeater.

-objects *object_list*

Specifies the set of objects to be associated with the topology_repeater. The objects may be hierarchical. Allowed object types are pins, ports, and cells.

DESCRIPTION

The **create_topology_repeater** command creates a topology_repeater on a topology_edge.

A topology_repeater will always have an owner edge, and once created cannot be transferred to a different edge. If the owner edge is deleted, then the repeaters owned by the edge are also deleted.

EXAMPLES

The following example creates a topology_repeater on the topology_edge "myedge", which itself resides in topology_plan "myplan":

```
prompt> create_topology_repeater -edge myedge
{myplan/myedge/TOPOLOGY_REPEATER0}
```

SEE ALSO

create_topology_repeater(2)
create_topology_edge(2)
get_topology_repeaters(2)
get_topology_edges(2)
remove_topology_repeaters(2)
remove_topology_edges(2)
report_topology_plans(2)

create_track

Creates the tracks for a routing or poly layer.

SYNTAX

collection **create_track**
-layer *layer*
[-space *track_pitch*]
[-count *number_of_tracks*]
[-coord *offset*]
[-width *wire_width*]
[-dir X | Y]
[-bbox *boundary_spec*]
[-reserved_for_width]
[-cell *cell*]
[-end_grid_low_offset *offset*]
[-end_grid_high_offset *offset*]
[-end_grid_low_steps *list_of_offsets*]
[-end_grid_high_steps *list_of_offsets*]
[-end_grid_relative_to core_area | block_boundary]
[-mask_pattern mask_one | mask_two]
[-relative_to core_area | block_boundary]
[-offset *offset*]

Data Types

layer string
track_pitch float
number_of_tracks integer
offset float
wire_width float
boundary_spec list
cell collection
list_of_offsets list
offset float

ARGUMENTS

-layer *layer*

Specifies the layer for which to create the tracks.

You can specify the layer name, layer number, or a collection that contains one layer object.

-space ***track_pitch***

Specifies the pitch distance between each track. i

The pitch is the center-to-center distance between two routing wires (the router places routes on top of each track). The value specified here is reported in the Pitch column of the **report_tracks** report.

By default, the distance is the same as the routing pitch from the physical library. The unit for the pitch distance is specified in the technology file; usually it is microns.

-count ***number_of_tracks***

Specifies the number of tracks to create.

By default, the tool fills the entire die area with tracks. If you specify the **-bbox** option, the tool fills the specified bounding box with tracks.

-coord ***start_x_or_y***

Specifies the location of the first track.

The x- and y-coordinates depend on the values specified with the **-dir** option. The units for the coordinates are specified in the technology file; usually it is microns.

By default, the track is located at half of the space distance inside of the die area.

-width ***wire_width***

Specifies the associated wire width attribute for rule-based wire tracks.

By default, wire tracks have no width attribute and have no usage constraint. Wire tracks with a width attribute can be used only by shapes whose width matches the track's width attribute.

-dir X | **Y**

Specifies the stepping direction in which the tracks are placed.

The **-dir X** option creates vertical tracks by stepping along the x-axis. The **-dir Y** option creates horizontal tracks by stepping along the y-axis.

By default, the stepping direction is derived from the routing direction of the layer. The stepping direction is X if the routing direction is vertical. The stepping direction is Y if the routing direction is horizontal.

-bbox ***boundary_spec***

Specifies the bounding boxes within which to create tracks.

The bounding box must fit the tracks of a given track object. For horizontal tracks, the first and last tracks must lie on the bottom and top edges of the box. For vertical tracks, the first and last tracks must lie on left and right edges of the box.

You can specify the bounding boxes directly by specifying a list of bounding boxes or you can specify polygons or geometric objects, which are decomposed into one or more bounding boxes.

To specify a list of bounding boxes, use the following syntax to specify the lower-left and upper-right corners of the bounding boxes:

{ {{*llx lly*} {*urx ury*}} ... }

To specify a list of polygons, use the following syntax to specify the vertices of the polygons:../../../../doc/nwtn/man/man2/create_track.2

{ {{*x y*} {*x y*} {*x y*} {*x y*} ...} ... }

The unit for the coordinates are specified in the technology file; usually it is microns.

Polygons can also be specified as the combined area of a heterogeneous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. For poly_rects, geo_masks, shapes, or other physical objects, the resulting area includes the areas of each object. In the case of layers, the resulting area includes the area of every shape on the layer.

-reserved_for_width

Specifies that the created tracks are reserved tracks. Reserved tracks can be used only by nets with a nondefault rule.

This option must be used with the **-width** option.

-cell *cell*

Specifies the physical cell in which to add the tracks.

The tracks are created in the cell's reference block using the coordinate system of the cell's top block. The cell must reference a block, not a library cell, unless you run this command in the library manager. The library manager can create tracks within library cells.

By default, the tracks are created in the current block.

-end_grid_low_offset *offset*

Specifies the offset from the left or bottom edge to the starting point of the first grid rectangle.

-end_grid_high_offset *offset*

Specifies the offset from the left or bottom edge to the ending point of the first grid rectangle.

-end_grid_low_steps *list_of_offsets*

Specifies a list of steps to use when creating subsequent grid points. Each offset in the list is the distance between the previous starting grid point and the current starting grid point.

-end_grid_high_steps *list_of_offsets*

Specifies a list of steps to use when creating subsequent grid points. Each offset in the list is the distance between the previous ending grid point and the current ending grid point.

-end_grid_relative_to core_area | block_boundary

Specifies that subsequent grid points need to be created with respect to core area bounding box or lower left of block bounding box. This option will also update values of options **-end_grid_low_offset** and **-end_grid_high_offset** as both of these options represent absolute value so providing **-end_grid_relative_to** option will recalculate them.

-mask_pattern mask_one | mask_two

Specifies the mask pattern for the created track.

-relative_to core_area | block_boundary

Specifies the alignment of track line. Based on the option, one of the track line will be aligned to lower boundary of core area bounding box or block bounding box. Although tracks will be created in complete block boundary. This option is mutually exclusive with **-coord** option.

-offset *start_x_or_y*

Specifies the offset of track line with respect to value provided in **-relative_to** option. First track line after start of core area or block boundary bounding box will be on a distance of value provided as part of the option. This option can be used only with **-relative_to** option.

DESCRIPTION

This command creates a group of tracks on the floorplan so the router can use them to perform detail routing. You must specify a routing layer for the tracks. The tracks can be saved in the Design Exchange Format (DEF) file or the design database file.

By default, this command creates a single track object. If the geometry specified by the **-bbox** option decomposes into multiple rectangles, the tool creates one track object per rectangle. The command returns a collection that contains the created track objects.

The **-end_grid_low_offset**, **-end_grid_high_offset**, **-end_grid_low_steps**, and **-end_grid_high_steps** options specify a nonuniform grid of allowed endpoints for wires. The low grid applies to left or bottom edges for horizontal or vertical tracks respectively. The high grid applies to right or top edges.

Wires can pass over these endpoints. If a wire terminates, the end of the wire must be on the grid. Grid points can be left untouched or uncovered by a wire.

For example, the following options describe a nonuniform grid:

```
-end_grid_low_offset 0
-end_grid_high_offset 2
-end_grid_low_steps {3,2}
-end_grid_high_steps {4,3}
```

And form the following grid points:

```
Low grid points are  x = 0, 3, 5, 8, 10, 13, 15
High grid points are x = 2, 6, 9, 13, 16, 20, 23
```

In this example, a wire can have a left edge at x = 10 and a right edge at x = 23. Another wire can have a left edge at x = 0 and a right edge at x = 6. These points form an abstract, nonuniform grid that constrains the router and specifies where wires can start and end.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates routing tracks for the M3 routing layer on the floorplan and reports the newly created routing tracks.

```
prompt> create_track -layer M3
Warning: Direction is not specified. Using the layer preferred
direction. (NDMUI-125)
Warning: Count value is not specified. Covering the bounding box of
track, depending on its space value. (NDMUI-136)
Warning: Coordinate value is not specified. Using coordinate that is
half of the space distance inside of the die area. (NDMUI-122)
Warning: Space is not specified. Using pitch. (NDMUI-124)
{TRACK_28}

prompt> report_tracks -layer M3
...
```

```
Layer        Direction    Start       Tracks   Pitch      Attr
--------------------------------------------------------------------------------
M3           Y            0.152       7333     0.304      default
1
```

The following example creates tracks within a rectilinear bounding box. The tool decomposes the bounding box into two rectangles and creates two track objects.

```
prompt> create_track -layer M3 \
  -bbox {{0 0} {20 0} {20 20} {10 20} {10 10} {0 10}}

prompt> report_tracks -layer M3
...
Layer        Direction    Start       Tracks   Pitch      Attr
--------------------------------------------------------------------------------
M3           Y            10.032      33       0.304      default
M3           Y            0.000       33       0.304      default
1

prompt> get_tracks -filter "layer_name==M3"
{TRACK_0 TRACK_1}
```

The following example uses the **-end_grid_*** options to specify grid points.

```
prompt> create_track -layer M3 -coord 0.96 -space 0.32 -dir Y \
  -width 0.32 -end_grid_low_offset 0.32 -end_grid_high_offset 0.96 \
  -end_grid_low_steps { 1.6 1.6 1.6 1.6 } \
  -end_grid_high_steps { 1.6 1.6 1.6 1.6 }

prompt> report_tracks -layer M3
...
Layer        Direction    Start       Tracks   Pitch
   Attr                   Edge_alignment
Low_offset   High_offset   Low_steps         High_steps
--------------------------------------------------------------------------------
M3           Y            0.960       6966     0.320
   non-reserved, width=0.320
0.320        0.960        1.600,1.600,1.600,1.600
                                1.600,1.600,1.600,1.600
1
```

The following example uses the **-end_grid_*** options along with **-end_grid_relative_to** to specify grid points.

```
prompt> get_attribute [get_core_area ] bbox
{10.5600 21.1200} {78.2000 79.9600}

prompt> create_track -layer M1 -dir Y -mask_pattern mask_one \
  -end_grid_high_steps {3 2 1} -end_grid_low_steps {1.5 2.5 4} \
  -end_grid_high_offset 4 -end_grid_low_offset 5 -end_grid_relative_to core_area

prompt> report_tracks -layer M1
...
Layer        Direction    Start       Tracks   Pitch
   Attr    Mask_pattern              Edge_alignment
Low_offset   High_offset   Low_steps         High_steps
--------------------------------------------------------------------------------
M1           Y            0.140       478      0.280
   default   mask_one
```

```
    7.560       2.560       1.500,2.500,4.000
                                3.000,2.000,1.000
    1
```

SEE ALSO

get_tracks(2)
remove_tracks(2)
report_tracks(2)

create_trunk

Creates route trunks for the specified nets on single layer in one direction.

SYNTAX

status **create_trunk**
-nets *objects*
-layer *layer*
[-direction horizontal | vertical]
[-width *width_list* | all]
[-num_wires *number*]
[-gravity driver | load | inout | all | percentage]
[-percentage *percentage*]
[-track_numbers *integer_list*]
[-track_coords *distance_list*]
[-track_step *step*]
[-track_ref_bbox *bounding_box*]
[-hierarchical]
[-shield_type full | low | high | cap_shield]
[-shield_nets *objects*]
[-shield_pullback *distance*]
[-avoid_obstructions]
[-do_not_check_shorts]
[-do_not_snap_to_track]
[-track_freedom *distance*]
[-start_pullback *distance*]
[-stop_pullback *distance*]
[-net_bbox_override *rect_list*]
[-of_cells *objects*]
[-exclude_cells *objects*]
[-create_obstruction]
[-transform *transformation*]
[-skip_lib_cells]
[-depth *depth_value*]
[-verbose]
[-quiet]

Data Types

objects collection
layer collection
width_list list of strings
number integer
percentage float
integer_list list of integers
distance_list list of floating-point numbers
step integer

bounding_box list
distance float
rect_list list of rectangles
transformation string
depth_value integer

ARGUMENTS

-nets *objects*

Specifies the group of nets or bundles for which to create route trunks. Bus bits are expanded left-to-right. The collection order is honored unless **-track_coords** or **-track_numbers** is specified which might force nets to be reordered.

-layer *layer*

Specifies the layer on which to create trunks.

-direction horizontal | vertical

Specifies the routing direction. Valid values are **horizontal** and **vertical**. By default, the direction is the preferred direction for the layer.

-width *width_list* | all

Specifies an unordered list of allowed track widths on which the tool can place a route trunk. Specify **-width all** to place route trunks on a track of any width. Nondefault routing rules take precedence over this option. By default, the command places route trunks only on tracks with the default width; tracks with a nondefault width are skipped.

The format of *width_list* is as follows:

-width width1
-width {width1 width2 ... }

If the *width_list* contains integer values, they are considered to be multiples of the default track width. If the *width_list* contains floating point values, they are considered to be absolute track widths. The following option and argument combinations are equivalent for a default track width of 0.25:

-width {1 2 3 4}
-width {0.25 0.5 0.75 1.0}

-num_wires *number*

Specifies the number of wires to create per net. Valid values are between 1 and 100000. When this option is specified, the command places the specified number of wires next to each other for the same net before proceeding to the next net in the list. By default, the command creates one wire per net.

-gravity driver | load | inout | all | percentage

Specifies the object type to use when aligning the trunks. Valid values are **driver**, **load**, **inout**, **all**, or **percentage**.

- *driver* - Align the trunk with the x- or y-coordinate of the pin of the driving pin
- *load* - Align the trunk with the x- or y-coordinate of the load pin
- *inout* - Align the trunk with the x- or y-coordinate of the inout pin
- *all* - Align the trunk with the center of the bounding box described by all the pins of the specified net

- *percentage* - Align the trunk based on percentage specified by the **-percentage** option. If the **-percentage** option is not specified, the default is 50% and the trunk aligns with the center the net bounding box.

The option is mutually exclusive with the **-track_coords** and **-track_numbers** options.

-percentage *percentage*

Aligns the route trunk based on a percentage of the bounding box for the net, starting from the left or bottom edge. The *percentage* value must be between 0 and 100 inclusive. A value of 0 places the trunk at the bottom or left edge of the bounding box, and a value of 100 places the trunk at the top or right edge of the bounding box. You must specify the **-gravity percentage** option together with the **-percentage** option. By default, the percentage is 50.

-track_numbers *integer_list*

Specifies the starting track number, or a list of specific track numbers, on which to place route trunks. Numbering is relative to the cell boundary origin or the origin of the bounding box specified by the **-track_ref_bbox** option. This option is mutually exclusive with the **-track_coords** and **-gravity** options.

To specify only the starting track, use **-track_numbers** together with a single integer value. For example, specify **-track_numbers 1** to begin placing route trunks at the first track within the boundary.

To specify a list of specific track numbers, use **-track_numbers** together with a list of integers. You must specify one integer value per net name as assigned by the **-nets** option. For example, specify **-track_numbers {1 3 5 7}** to place the route trunk on the first, third, fifth, and seventh tracks within the boundary when four nets are specified with the **-nets** option. If you specify the track numbers not in increasing order, the tool creates the route trunks but issues a LED-110 warning message.

-track_coords *distance_list*

Specifies a single x- or y-coordinate, or a list of x- or y-coordinates, on which to start placing the route trunk. The list must contain one value per net name as assigned by the **-nets** option. The trunk is snapped to the closest track unless **-do_not_snap_to_track** is specified. The option is mutually exclusive with the **-track_numbers** option. The option can be used with the **-track_step** option and the list should contain only one element that specifies the starting coordinate.

If the track is obstructed, the command issues an LED-110 warning message, skips the net, and does not create a route trunk in the obstructed track.

-track_step *step*

Specifies the step to use when placing routes for the remaining nets. The first track is placed as specified by the **-track_numbers** option or the **-track_coords** option. Successive tracks are placed according to the *step* value. You can specify a negative value to place successive tracks in the opposite direction. By default, the step is one and vertical routes are placed in consecutive tracks to the right of the first route trunk and horizontal routes are placed in consecutive tracks above the first route trunk.

-track_ref_bbox *bounding_box*

Specifies the bounding box on which to start track counting. If the bounding box is too small to contain all the route trunks, the tool issues an LED-110 message, skips the nets outside the boundary, and does not create the routes for those nets.

-hierarchical

Traces the net through the hierarchy and uses connections of all traced nets to determine the bounding box for the net for the user-specified topology.

-shield_type full | low | high | cap_shield

Specifies shielding types. Valid values are **full**, **low**, **high**, **cap_shield** as follows:

- *full* : Shield each net on both sides
- *low* : Shield each net on low side

- *high* : Shield each net on high side
- *cap_shield* : Shield first net on low side and last net on high side

-shield_nets *objects*

Specifies the shielding net pattern. The nets in shielding pattern are used cyclically.

-shield_pullback *distance*

Specifies the value in microns to cut or extend the shield ends.

-avoid_obstructions

Finds the next available, unobstructed routing track to create the trunk. The command issues a warning message if the trunk is placed outside the net bounding box. Use the **-track_freedom** option to expand the region where trunks can be shifted without warnings.

By default, if an obstruction is found, the command issues an LED-110 warning message, skips the net, and does not create a route trunk in the obstructed track.

-do_not_check_shorts

Creates route trunks that might create a short circuit with an existing route and overlap route obstructions. Use this option for a quick estimation and preview of route trunk placement without checking for shorts and obstructions.

-do_not_snap_to_track

Avoids snapping to a legal track. By default, route trunks are snapped to tracks.

-track_freedom *distance*

Expands the region where trunks could be shifted without issuing a warning message. This option allows route trunks to be created outside the boundary specified by the **-track_ref_bbox** option or the bounding box defined by the pin connections for the specified nets. You must specify the **-avoid_obstructions** option together with the **-track_freedom** option. By default, tracks are placed within the boundary.

-start_pullback *distance*

Specifies the distance in microns to shorten or extend the trunk at the starting point derived from net bbox (left or bottom). The positive *distance* value will shorten the segment, while the negative *distance* value will extend the segment.

-stop_pullback *distance*

Specifies the distance in microns to shorten or extend the trunk at the stop point derived from net bbox (right or top). The positive *distance* value will shorten the segment, while the negative *distance* value will extend the segment.

-net_bbox_override *rect_list*

Specifies the net bounding box to use when creating the trunk. The actual bounding box for the net is ignored. The bounding box specified by this option is applied to all nets.

-of_cells *cells*

Specifies the cells to use to calculate the net bounding box. The option can be used in multi-fanout scenarios to cut unnecessary branches.

-exclude_cells *cells*

Excludes the specified cells when calculating the net bounding box. This option can be used on nets with multiple fanouts to remove unnecessary branches.

-create_obstruction

Creates a routing blockage instead of creating a route shape. The routing blockage is similar to a blockage created with the Create Routing Blockage tool or the **create_routing_blockage** command.

-transform *transformation*

Applies the transformation expression to the route trunks. The transformation expression is one or two transformation steps that are applied to the route trunk. A transformation step is a move by a specific x- and y-distance or an orientation change. The command honors routing obstructions in the new location.

To create the route trunk and move it, specify **-transform {x y}** where x and y are the x- and y-distances in microns to move the trunks. To change the orientation of the route trunk, specify **-transform {orientation}**, where orientation is one of MX, MXR90, MY, MYR90, R0, R180, R270, and R90.

To combine two transformations, specify the delta and orientation transformations in the same list. For example, to specify that the route trunk is rotated 270 degrees and moved to the right by 10 microns, specify **-transform {{R270} {10 0}}**.

-skip_lib_cells

Do not descend into library cells to search for obstructions. This option must be used together with the **-avoid_obstructions** option. By default, the command checks for obstructions within library cells.

-depth *depth_value*

Limits the level of hierarchy to search for obstructions. Specify a *depth_value* of zero to search only the top level. This option must be used together with the **-avoid_obstructions** option. By default, all levels of hierarchy are searched.

-verbose

Prints detailed information about warning and errors.

-quiet

Suppresses all warning messages.

DESCRIPTION

This command creates trunks for the specified nets on a single layer in one direction. The command can be used for straight connections for critical signals where traditional routers might struggle. Although it is not a router in classical terms, the command can still ensure clean routing by shifting the routes.

The command supports the following operational modes:

- **Skip** mode (default): The trunks that cannot be routed are skipped and reported using **-verbose** option.
- **Dirty** mode: No checking on trunks which create violations with existing routes. Specify this mode with the **-do_not_check_shorts** option.
- **Clean** mode: In this case, no trunks should violate spacing. Specify this mode with the **-avoid_obstruction** option.

The command supports the following track snapping modes:

- Default tracks: Trunks should always snap to tracks and avoid overlaps with existing routes. If an overlap occurs, snap to the next available track by spreading the wires.
- Nondefault tracks: Specify this mode with the **-width** option.

- Trackless: Tracks are ignored and routes are created within the minimum spacing distance. Specify this mode with the **-do_not_snap_to_track** option.

The width of a given route trunk is width of the track on which it is placed, or the default track width if the **-do_not_snap_to_track** option is specified. If a net has a non-default routing rule (NDR) that requires a particular width on the layer, the route width is the NDR width and the NDR width takes precedence over the width specified by the **-width** argument. The spacing between route trunks can also be adjusted using NDR spacing. To adjust shielding, set the NDR shielding width and spacing.

The command supports multiple levels of physical hierarchy, can operate on a net within any block,and can create the corresponding routes within that block.

EXAMPLES

The following example creates trunks on vertical layer M2 starting from coordinate 8.4 with step of 2 and uses nonuniform track widths 0.4 and 0.6.

```
prompt> create_trunk -nets bus1 -layer M2 -track_coords {8.4} \
  -track_step 2 -width {0.4 0.6}
```

The following example creates route trunks on the M3 horizontal layer for four nets. The net1 net is placed on the third track above the bounding box defined by the **-track_ref_bbox** option. Successive route trunks are placed on every other track.

```
prompt> create_trunk -nets {net1 net2 net3 net4} -layer M3 -track_numbers 3 \
  -track_step 2 -track_ref_bbox {{100 100} {500 500}}
```

The following example creates route trunks on the M3 horizontal layer and moves them 40 microns in the x-direction and 10 microns in the y-direction. If obstructions exist in the new location, the route trunks are placed on the next available routing track.

```
prompt> create_trunk -nets {net1 net2 net3 net4} -layer M3 \
  -transform {40 10} -avoid_obstructions
```

SEE ALSO

create_routing_blockage(2)
create_trunk_pin_to_pin(2)
create_trunk_pin_to_trunk(2)
push_down_objects(2)
report_tracks(2)

create_trunk_pin_to_pin

Creates trunks from a pin or port to another pin or port. Nets can have one or more fanouts.

SYNTAX

status **create_trunk_pin_to_pin**
-nets *objects*
-layer_legs *layer_list* | all
-extensions *distance_list*
[-start_from low_x | low_y | high_x | high_y]
[-order ascending | descending]
[-hierarchical]
[-shield_type low | high | full | cap_shield]
[-shield_nets *objects*]
[-shield_pullback *distance*]
[-avoid_obstructions]
[-do_not_check_shorts]
[-do_not_snap_to_track]
[-do_not_add_vias]
[-horizontal_freedom *distance*]
[-vertical_freedom *distance*]
[-start_pullback *distance*]
[-stop_pullback *distance*]
[-net_bbox_override *rect_list*]
[-of_cells *objects*]
[-exclude_cells *objects*]
[-create_obstruction]
[-skip_lib_cells]
[-depth *depth_value*]
[-verbose]
[-quiet]

Data Types

objects collection
layer_list list of strings
distance_list list of floating point numbers
distance float
rect_list list of rectangles
depth_value int

ARGUMENTS

-nets *objects*

Specifies the group of nets or bundles. The collection order is honored unless **-order** is specified which forces nets to be reordered.

-layer_legs *layer_list* | all

Specifies the layers for which to create segments. The tracks with default width are used by default. You can specify the list of track widths on which to place routes by using the following syntax:

-layer_legs {{{<layer1> {<width1> <width2> ... }} ... }

To use all tracks on a layer, specify **all** as follows:

-layer_legs {{{<layer1> all} ... }

-extensions *list*

Specifies the distance which to extend each segment. Extensions can be specified as a fraction of the net bounding box, a percentage of the net bounding box, in microns, or as user-defined units. The extensions list corresponds to a list of layer segments.

-start_from low_x | low_y | high_x | high_y

Specifies the relative pin location to start from. Segment lengths or percentages specified by the **-extensions** option are applied from the starting pin. Valid values are **low_y**, **low_x**, **high_y**, **high_x** as follows:

- **low_y** : Start from the bottommost pin (default)
- **low_x** : Start from the leftmost pin
- **high_y** : Start from the topmost pin
- **high_x** : Start from the rightmost pin

-order ascending | descending

Specifies the route ordering used when creating new orthogonal route segments. Valid values are **ascending** or **descending** as follows:

- **ascending** : Sort nets in ascending order at the turn based on the startpoint coordinates
- **descending** : Sort nets in descending order at the turn based on the startpoint coordinates

By default, the user-specified net or bundle order is preserved.

-hierarchical

Traces the net through the hierarchy and uses connections of all traced nets to determine the bounding box for the net for the user-specified topology.

-shield_type low | high | full | cap_shield

Specifies the shielding type. Valid values are **full**, **low**, **high**, **cap_shield** as follows:

- **low** : Shield each net on the low side
- **high** : Shield each net on the high side
- **full** : Shield each net on the both sides
- **cap_shield** : Shield the first net on the low side and last net on the high side

-shield_nets *objects*

Specifies the shielding net pattern. The nets in the shielding pattern are used cyclically.

-shield_pullback *distance*

Specifies the micron value to cut or extend the shield ends.

-avoid_obstructions

Finds the next free, unobstructed routing track to create the trunk. The command issues a warning message if the trunk is placed outside the net bounding box. Use the **-horizontal_freedom** and **-vertical_freedom** options expand the region where trunks could be shifted without creating a warning message.

By default, if an obstruction is found, the command skips the net and issues an LED-110 warning message.

-do_not_check_shorts

Avoids searching for obstructions. The option can be used for quick estimation and preview.

-do_not_snap_to_track

Avoids snapping the trunks to legal tracks.

-do_not_add_vias

Avoids adding vias. The option can be used if you are planning to insert clean vias using some other command as at the moment **create_trunk_pin_to_pin** does not check cut to cut via spacings. Please note that via metal enclosures are included in all spacing checks performed on trunks.

-horizontal_freedom *distance*

Expands the region horizontally where trunks could be shifted without issuing a warning message. The *distance* is in microns. This option allows route trunks to be created outside the boundary specified by the **-net_bbox_override** option. By default, the *distance* is zero, routes are created within the boundary, and nets are skipped if they do not fit in the bounding box.

-vertical_freedom *distance*

Expands the region vertically where trunks could be shifted without issuing a warning message. The *distance* is in microns. This option allows route trunks to be created outside the boundary specified by the **-net_bbox_override** option. By default, the *distance* is zero, routes are created within the boundary, and nets are skipped if they do not fit in the bounding box.

-start_pullback *distance*

Specifies the distance in microns to shorten or extend the trunk at the starting pin. Specify a positive *distance* value to shorten the segment and prevent it from connecting to the pin. Specify a negative *distance* value to extend the segment beyond the pin.

-stop_pullback *distance*

Specifies the distance in microns to shorten or extend the trunk at the ending pin. Specify a positive *distance* value to shorten the segment and prevent it from connecting to the pin. Specify a negative *distance* value to extend the segment beyond the pin.

-net_bbox_override *rect_list*

Specifies the net bounding box to use when creating the trunk. The actual bounding box for the net is ignored. The bounding box specified by this option is applied to all nets.

-of_cells *objects*

Specifies the cells to use to calculate the net bounding box. The option can be used in multi-fanout scenarios to cut unnecessary branches.

-exclude_cells *objects*

Excludes the specified cells when calculating the net bounding box. The option can be used in multi-fanout scenarios to cut unnecessary branches.

-create_obstruction

Creates a routing obstruction instead of path objects.

-skip_lib_cells

Do not descend into library cells to search for obstructions. This option must be used together with the **-avoid_obstructions** option.

-depth *depth_value*

Specifies the hierarchy depth to search for obstructions. By default, all levels of hierarchy are searched. This option must be used together with the **-avoid_obstructions** option.

-verbose

Print detailed information about warning and errors.

-quiet

Suppress all warnings

DESCRIPTION

This command creates trunks from one pin to another pin. The net can have one or more fanouts. The command can be used to create a straight connection for critical signals where traditional routers might struggle. Although it is not a router in classical terms, it can still ensure clean routing by shifting the routes. The command supports different routing topologies such as L, Z, U, and so on. Specify the topologies with the **-layer_legs** and **-extensions** options. The routing or wiring directions are determined primarily by the **-start_from** option.

The command supports the following operational modes:

- **Skip** mode (default): The trunks that cannot be routed are skipped and reported using **-verbose** option.
- **Dirty** mode: No checking on trunks which create violations with existing routes. Specify this mode with the **-do_not_check_shorts** option.
- **Clean** mode: In this case, no trunks should violate spacing. Specify this mode with the **-avoid_obstruction** option.

The command supports the following track snapping modes:

- Default tracks: Trunks should always snap to tracks without overlapping, if an overlap would occur, snap to the next available track by spreading the wires.
- Non default tracks: Specify this mode with the **-layer_legs {{{<layer1> {<width1> <width2> ... }} ... }** option.
- Trackless: Tracks are ignored and routes are created within the minimum spacing distance. Specify this mode with the **-do_not_snap_to_track** option.

The trunk routes derive width from underlying tracks or use the default width if **-do_not_snap_to_track** is specified. If the net has a non-default routing rule (NDR) that requires a width on the layer, the route width is set according NDR width and that width takes precedence over a **-layer_legs** argument for track lookup. The spacing between trunk routes can also be adjusted using NDR spacing. To adjust shielding user can set NDR shielding width and spacing.

The command supports multiple levels of physical hierarchy and can operate on a net within any block of a design and create the

corresponding routes within that block.

The command also supports cross-hierarchy routing for top-down flows. When the **-hierarchical** option is specified, the command traces the net across the hierarchy and uses connections of all traced nets to determine the bounding box for the net for the user-specified topology. All routes created would belong to a single top-level net. To finalize the routing, you must use the **push_down_objects** command to push the top-level routing into blocks the route intersects.

EXAMPLES

The following example creates an L-topology that connects pins on vertical layer M2 and horizontal layer M3. Each segment extends to fill 100% of net bounding box.

```
prompt> create_trunk_pin_to_pin -layer_legs {M2 M3} \
  -extensions {1 1}
```

The following example create the same route as the previous example but uses percentages to specify the extensions.

```
prompt> create_trunk_pin_to_pin -nets bus1 -layer_legs {M2 M3} \
  -extensions {100% 100%}
```

The following example uses an alternate syntax to specify absolute distances with either **u** or **d** suffixes:

```
prompt> create_trunk_pin_to_pin -nets bus1 -layer_legs {M2 M3} \
  -extensions {25.00u 25.00d}
```

The following example creates an L-topology route with track widths of 0.4 and 0.6 on layer M2. Routes on layer M3 can be created on tracks of any width.

```
prompt> create_trunk_pin_to_pin -nets bus1 \
  -layer_legs {{{M2 {0.4 0.6}} {M3 all}} -extensions {1 1}
```

SEE ALSO

create_trunk(2)
create_trunk_pin_to_trunk(2)
push_down_objects(2)

create_trunk_pin_to_trunk

Creates trunks from pins and ports to existing route trunks for the specified nets or bundles.

SYNTAX

status **create_trunk_pin_to_trunk**
-nets *objects*
[-layer *layer*]
[-direction horizontal | vertical]
[-width *width*]
[-min_layer *layer*]
[-cluster_proximity *distance*]
[-hierarchical]
[-shield_type full | low | high | cap_shield]
[-shield_nets *objects*]
[-shield_pullback *distance*]
[-avoid_obstructions]
[-do_not_check_shorts]
[-do_not_snap_to_track]
[-do_not_add_vias]
[-horizontal_freedom *distance*]
[-vertical_freedom *distance*]
[-pullback *distance*]
[-of_cells *objects*]
[-exclude_cells *objects*]
[-create_obstruction]
[-skip_lib_cells]
[-depth *depth_value*]
[-verbose]
[-quiet]

Data Types

objects collection
layer collection
width collection
distance float
depth_value integer

ARGUMENTS

-nets *objects*

Specifies the group of nets or bundles to connect to pins and ports. The collection order is honored.

-layer *layer*

Specifies the layer for which to create trunks. If this option is omitted, the trunk layers are derived from pins and ports.

-direction horizontal | vertical

Specifies the routing direction. Valid values are **horizontal** and **vertical**. By default, the direction is the preferred direction for the layer.

-width *width*

Specifies the width of the routes to create. A float value specifies the absolute width of the routes. An integer value is multiplied by the pin width to determine the route width.

-min_layer *layer*

Specifies the minimum layer on which pin and port trunks can be routed. By default, the terminal layer is used unless it is lower than minimum layer.

-cluster_proximity *distance*

Specifies the range in microns within which to group individual pin-to-trunk route shapes to a single route shape. The command creates the route segment from the trunk, but might not complete the connection to the pin. By default, the *distance* is zero and the command connects each pin on the net to the trunk by its own route shape.

Pins are clustered together if the following conditions are met:

- Pins can be connected with the same wire
- Pins are located on the same side relative to the wire
- Pin projections to the wire is within the specified threshold distance

-hierarchical

Traces the net through the hierarchy and uses connections to all traced nets to determine the bounding box for the net for the user-specified topology.

-shield_type full | low | high | cap_shield

Specifies shielding types. Valid values are **full**, **low**, **high**, **cap_shield** as follows:

- **full** : shield each net on both sides
- **low** : shield each net on low side
- **high** : shield each net on high side
- **cap_shield** : shield first net on low side and last net on high side

-shield_nets *objects*

Specifies the shielding net pattern. The nets in shielding pattern are used cyclically.

-shield_pullback *distance*

Specifies the value in microns to cut or extend the shield ends.

-avoid_obstructions

Finds the next available, unobstructed routing track to create the trunk. The command issues a warning message if the trunk is placed outside the net bounding box. Use the **-horizontal_freedom** and **-vertical_freedom** options to expand the region where the command can shift trunks without issuing a warning message.

By default, if an obstruction is found, the command issues an LED-110 warning message, skips the net, and does not create a route trunk in the obstructed track.

-do_not_check_shorts

Ignores searching for obstructions. The option can be used for quick estimation and preview.

-do_not_snap_to_track

Avoids snapping to a legal track.

-do_not_add_vias

Avoids adding vias. The option can be used if you are planning to insert clean vias using some other command as at the moment **create_trunk_pin_to_trunk** does not check cut to cut via spacings. Please note that via metal enclosures are included in all spacing checks performed on trunks.

-horizontal_freedom *distance*

Expands the region where trunks could be shifted without warnings.

-vertical_freedom *distance*

Expands the region where trunks could be shifted without warnings.

-pullback *distance*

Specifies the distance in microns to cut or extend the trunks from pins and ports.

-of_cells *cells*

Specifies the cells to use to calculate the net bounding box. This option can be used for nets with large fanout to reduce the number of branches.

-exclude_cells *cells*

Excludes the specified cells when calculating the net bounding box. The option can be used for nets with large fanout to reduce the number of branches.

-create_obstruction

Creates a routing blockage instead of creating a route shape. The routing blockage is similar to a blockage created with the Create Routing Blockage tool or the **create_routing_blockage** command.

-skip_lib_cells

Do not descend into library cells to search for obstructions. This option must be used together with the **-avoid_obstructions** option.

-depth *depth_value*

Limits the level of hierarchy to search for obstructions. Specify a *depth_value* of zero to search only the top level. This option must be used together with the **-avoid_obstructions** option. By default, all levels of hierarchy are searched.

-verbose

Prints detailed information about warning and errors.

-quiet

Suppresses all warnings

DESCRIPTION

This command creates trunks for the specified nets on the layers derived from the pin or port shape for the net. The command can be used for straight connections for critical signals where traditional routers might struggle. Although it is not a router in classical terms, the command can still ensure clean routing by shifting the routes.

The command supports the following operational modes:

- **Skip** mode (default): The trunks that cannot be routed are skipped and reported using **-verbose** option.
- **Dirty** mode: No checking on trunks which create violations with existing routes. Specify this mode with the **-do_not_check_shorts** option.
- **Clean** mode: In this case, no trunks should violate spacing. Specify this mode with the **-avoid_obstructions** option.

The command supports the following track snapping modes:

- Default tracks: Trunks should always snap to tracks without overlapping, if an overlap would occur, snap to the next available track by spreading the wires.
- Non default tracks: Specify this mode with the **-width {<width1> <width2> ... }** option.
- Trackless: Tracks are ignored and routes are created within the minimum spacing distance. Specify this mode with the **-do_not_snap_to_track** option.

The width of the trunk routes is derived from the track width or the default width if the **-do_not_snap_to_track** option is specified. If a net has a nondefault routing rule (NDR) that requires a width on the layer, then the route width is set according to the NDR width and the NDR width takes precedence over the **-width** argument when assigning a track for the route. The spacing between trunk routes can also be adjusted using NDR spacing. To adjust shielding, set the NDR shielding width and spacing.

The command supports multiple levels of physical hierarchy and can operate on a net within any block of a design and create the corresponding routes within that block.

EXAMPLES

The following example creates trunks on vertical layer M2 starting from pins/ports of a specified set of nets to existing trunks. It uses nonuniform track widths 0.4 and 0.6.

```
prompt> create_trunk_pin_to_trunk -nets bus1 -layer M2 -width {0.4 0.6}
```

SEE ALSO

create_trunk(2)
create_trunk_pin_to_pin(2)
push_down_objects(2)

create_trunk_shared_track

Creates route trunks for the specified nets on same track in one direction.

SYNTAX

status **create_trunk_shared_track**
-nets *objects*
-layer *layer*
[-direction horizontal | vertical]
[-width *width_list* | all]
[-track_coord *distance*]
[-track_number *integer*]
[-track_ref_bbox *bounding_box*]
[-shield_type full | low | high | cap_shield]
[-shield_nets *objects*]
[-shield_pullback *distance*]
[-avoid_obstructions]
[-do_not_check_shorts]
[-do_not_snap_to_track]
[-track_freedom *distance*]
[-start_pullback *distance*]
[-stop_pullback *distance*]
[-gap *number*]
[-net_bbox_override *rect_list*]
[-of_cells *objects*]
[-exclude_cells *objects*]
[-create_obstruction]
[-skip_lib_cells]
[-depth *depth_value*]
[-verbose]
[-quiet]

Data Types

objects collection
layer collection
width_list list of strings
number integer
step integer
bounding_box list
distance float
rect_list list of rectangles
depth_value integer

ARGUMENTS

-nets *objects*

Specifies the group of nets for which to share the same track. The collection is reordered automatically based on net bounding boxes. It is assumed that bounding boxes of nets do not overlap or overlap a little. The typical scenario is creation of trunks for a set of net separated by repeaters.

-layer *layer*

Specifies the layer on which to create trunks.

-direction horizontal | vertical

Specifies the routing direction. Valid values are **horizontal** and **vertical**. By default, the direction is the preferred direction for the layer.

-width *width_list* | all

Specifies an unordered list of allowed track widths on which the tool can place a route trunk. Specify **-width all** to place route trunks on a track of any width. Nondefault routing rules take precedence over this option. By default, the command places route trunks only on tracks with the default width; tracks with a nondefault width are skipped.

The format of *width_list* is as follows:

-width width1
-width {width1 width2 ... }

If the *width_list* contains integer values, they are considered to be multiples of the default track width. If the *width_list* contains floating point values, they are considered to be absolute track widths. The following option and argument combinations are equivalent for a default track width of 0.25:

-width {1 2 3 4}
-width {0.25 0.5 0.75 1.0}

-track_coord *distance*

Specifies a single x- or y-coordinate on which to start placing the route trunk. The trunk is snapped to the closest track unless **-do_not_snap_to_track** is specified. The option is mutually exclusive with the **-track_number** option.

If the track is obstructed, the command issues an LED-110 warning message, skips the net, and does not create a route trunk in the obstructed track.

-track_number *number*

Specifies the track number on which to place route trunks. Numbering is relative to the cell boundary origin or the origin of the bounding box specified by the **-track_ref_bbox** option. This option is mutually exclusive with the **-track_coord**.

-track_ref_bbox *bounding_box*

Specifies the bounding box on which to start track counting. If the bounding box is too small to contain all the route trunks, the tool issues an LED-110 message, skips the nets outside the boundary, and does not create the routes for those nets.

-shield_type full | low | high | cap_shield

Specifies shielding types. Valid values are **full**, **low**, **high**, **cap_shield** as follows:

- *full* : Shield each net on both sides

- *low* : Shield each net on low side
- *high* : Shield each net on high side
- *cap_shield* : Shield first net on low side and last net on high side

-shield_nets *objects*

Specifies the shielding net pattern. The nets in shielding pattern are used cyclically.

-shield_pullback *distance*

Specifies the value in microns to cut or extend the shield ends.

-avoid_obstructions

Finds the next available, unobstructed routing track to create the trunk. The command issues a warning message if the trunk is placed outside the net bounding box. Use the **-track_freedom** option to expand the region where trunks can be shifted without warnings.

By default, if an obstruction is found, the command issues an LED-110 warning message, skips the net, and does not create a route trunk in the obstructed track.

-do_not_check_shorts

Creates route trunks that might create a short circuit with an existing route and overlap route obstructions. Use this option for a quick estimation and preview of route trunk placement without checking for shorts and obstructions.

-do_not_snap_to_track

Avoids snapping to a legal track. By default, route trunks are snapped to tracks.

-track_freedom *distance*

Expands the region where trunks could be shifted without issuing a warning message. This option allows route trunks to be created outside the boundary specified by the **-track_ref_bbox** option or the bounding box defined by the pin connections for the specified nets. You must specify the **-avoid_obstructions** option together with the **-track_freedom** option. By default, tracks are placed within the boundary.

-start_pullback *distance*

Specifies the distance in microns to shorten or extend the trunk at the starting point derived from net bbox (left or bottom). The positive *distance* value will shorten the segment, while the negative *distance* value will extend the segment.

-stop_pullback *distance*

Specifies the distance in microns to shorten or extend the trunk at the stop point derived from net bbox (right or top). The positive *distance* value will shorten the segment, while the negative *distance* value will extend the segment.

-gap *number*

Specifies spacing multiplier to put a gap between trunks on a track.

-net_bbox_override *rect_list*

Specifies the net bounding box to use when creating the trunk. The actual bounding box for the net is ignored. The bounding box specified by this option is applied to all nets.

-of_cells *cells*

Specifies the cells to use to calculate the net bounding box. The option can be used in multi-fanout scenarios to cut unnecessary branches.

-exclude_cells *cells*

Excludes the specified cells when calculating the net bounding box. This option can be used on nets with multiple fanouts to remove unnecessary branches.

-create_obstruction

Creates a routing blockage instead of creating a route shape. The routing blockage is similar to a blockage created with the Create Routing Blockage tool or the **create_routing_blockage** command.

To create the route trunk and move it, specify **-transform {x y}** where x and y are the x- and y-distances in microns to move the trunks. To change the orientation of the route trunk, specify **-transform {orientation}**, where orientation is one of MX, MXR90, MY, MYR90, R0, R180, R270, and R90.

To combine two transformations, specify the delta and orientation transformations in the same list. For example, to specify that the route trunk is rotated 270 degrees and moved to the right by 10 microns, specify **-transform {{R270} {10 0}}**.

-skip_lib_cells

Do not descend into library cells to search for obstructions. This option must be used together with the **-avoid_obstructions** option. By default, the command checks for obstructions within library cells.

-depth *depth_value*

Limits the level of hierarchy to search for obstructions. Specify a *depth_value* of zero to search only the top level. This option must be used together with the **-avoid_obstructions** option. By default, all levels of hierarchy are searched.

-verbose

Prints detailed information about warning and errors.

-quiet

Suppresses all warning messages.

DESCRIPTION

This command creates trunks for the specified nets on a single layer in one direction. The command can be used for straight connections for critical signals where traditional routers might struggle. Although it is not a router in classical terms, the command can still ensure clean routing by shifting the routes.

The command supports the following operational modes:

- **Skip** mode (default): The trunks that cannot be routed are skipped and reported using **-verbose** option.
- **Dirty** mode: No checking on trunks which create violations with existing routes. Specify this mode with the **-do_not_check_shorts** option.
- **Clean** mode: In this case, no trunks should violate spacing. Specify this mode with the **-avoid_obstruction** option.

The command supports the following track snapping modes:

- Default tracks: Trunks should always snap to tracks and avoid overlaps with existing routes. If an overlap occurs, snap to the next available track by spreading the wires.
- Nondefault tracks: Specify this mode with the **-width** option.
- Trackless: Tracks are ignored and routes are created within the minimum spacing distance. Specify this mode with the **-**

do_not_snap_to_track option.

The width of a given route trunk is width of the track on which it is placed, or the default track width if the **-do_not_snap_to_track** option is specified. If a net has a non-default routing rule (NDR) that requires a particular width on the layer, the route width is the NDR width and the NDR width takes precedence over the width specified by the **-width** argument. The spacing between route trunks can also be adjusted using NDR spacing. To adjust shielding, set the NDR shielding width and spacing.

The command supports multiple levels of physical hierarchy, can operate on a net within any block,and can create the corresponding routes within that block.

EXAMPLES

The following example creates trunks for four nets on vertical layer M2. The trunks are placed on nonuniform track of a given widths (0.4 or 0.6) nearest to coordinate 8.4.

```
prompt> create_trunk_shared_track -nets {net1 net2 net3 net4} \
    -layer M2 -track_coord {8.4} -width {0.4 0.6}
```

SEE ALSO

create_routing_blockage(2)
create_trunk(2)
create_trunk_pin_to_pin(2)
create_trunk_pin_to_trunk(2)
push_down_objects(2)
report_tracks(2)

create_trunk_topology

Creates trunks from a topology edge.

SYNTAX

status **create_trunk_topology**
[-plan *plan*]
[-edge *edge*]
[-shield_type low | high | full | cap_shield]
[-shield_nets *objects*]
[-shield_pullback *distance*]
[-avoid_obstructions]
[-do_not_check_shorts]
[-do_not_snap_to_track]
[-do_not_add_vias]
[-horizontal_freedom *distance*]
[-vertical_freedom *distance*]
[-start_pullback *distance*]
[-stop_pullback *distance*]
[-skip_lib_cells]
[-verbose]
[-quiet]

Data Types

edge name or collection
objects names or collection
distance float

ARGUMENTS

-plan *object*

Specifies the topology plan name or collection to create trunks for. This option is mutually exclusive with **-edge** option.

-edge *object*

Specifies the topology edge name or collection to create trunks for. This option is mutually exclusive with **-plan** option. There are currently two types of edges that are fully supported.

- Both edge nodes have explicit or implicit associations with pins and **shape_layers** attribute specifies rectilinear path between them.
- One of the nodes is tap node associated with another edge, while other node is associated with pins. Only straight connection is currently possible for such an edge.

-shield_type low | high | full | cap_shield

Specifies the shielding type. Valid values are **full**, **low**, **high**, **cap_shield** as follows:

- **low** : Shield each net on the low side
- **high** : Shield each net on the high side
- **full** : Shield each net on the both sides
- **cap_shield** : Shield the first net on the low side and last net on the high side

-shield_nets *objects*

Specifies the shielding net pattern. The nets in the shielding pattern are used cyclically.

-shield_pullback *distance*

Specifies the micron value to cut or extend the shield ends.

-avoid_obstructions

Finds the next free, unobstructed routing track to create the trunk. The command issues a warning message if the trunk is placed outside the net bounding box. Use the **-horizontal_freedom** and **-vertical_freedom** options expand the region where trunks could be shifted without creating a warning message.

By default, if an obstruction is found, the command skips the net and issues an LED-110 warning message.

-do_not_check_shorts

Avoids searching for obstructions. The option can be used for quick estimation and preview.

-do_not_snap_to_track

Avoids snapping the trunks to legal tracks.

-do_not_add_vias

Avoids adding vias. The option can be used if you are planning to insert clean vias using some other command as at the moment **create_trunk_topology** does not check cut to cut via spacings. Please note that via metal enclosures are included in all spacing checks performed on trunks.

-horizontal_freedom *distance*

Expands the region horizontally where trunks could be shifted without issuing a warning message. The *distance* is in microns. This option allows route trunks to be created outside the boundary specified by the **-net_bbox_override** option. By default, the *distance* is zero, routes are created within the boundary, and nets are skipped if they do not fit in the bounding box.

-vertical_freedom *distance*

Expands the region vertically where trunks could be shifted without issuing a warning message. The *distance* is in microns. This option allows route trunks to be created outside the boundary specified by the **-net_bbox_override** option. By default, the *distance* is zero, routes are created within the boundary, and nets are skipped if they do not fit in the bounding box.

-start_pullback *distance*

Specifies the distance in microns to shorten or extend the trunk at the starting pin. Specify a positive *distance* value to shorten the segment and prevent it from connecting to the pin. Specify a negative *distance* value to extend the segment beyond the pin.

-stop_pullback *distance*

Specifies the distance in microns to shorten or extend the trunk at the ending pin. Specify a positive *distance* value to shorten the segment and prevent it from connecting to the pin. Specify a negative *distance* value to extend the segment beyond the pin.

-skip_lib_cells

Do not descend into library cells to search for obstructions. This option must be used together with the **-avoid_obstructions** option.

-verbose

Print detailed information about warning and errors.

-quiet

Suppress all warnings

DESCRIPTION

The command creates trunks for a topology edge. It can be used to create a straight connection for critical signals where traditional routers might struggle. Although it is not a router in classical terms, it can still ensure clean routing by shifting the routes. The command supports different routing topologies such as L, Z, U, and so on.

This command is slightly different semantically compared to sibling command **create_trunk_pin_to_pin** as it supports corner styles and specification of reference bit. It also has fewer options as many important arguments are derived from a single **-edge** option. Here is the list of implicitly derived options:

- Nets and supernets or bundles of nets and supernets are derived from a topology edge, topology plan or pins of a start topology node.
- Routing layers and distances have to be specified as topology edge attribute **shape_layers**
- Start and end pins simply correspond to start and end node of a specified edge
- Reference bit alignment is derived from **alignment** attribute of start topology node. By default the alignment is **center** and attribute **alignment** is unset. User can set it to **left** or **right**. It defines routing bit order and shifting directions.
- The bus corner style is derived from **corner_style** attribute of start topology node. By default the corner type is **cross** and it can also be set to **river**

The command supports the following operational modes:

- **Skip** mode (default): The trunks that cannot be routed are skipped and reported using **-verbose** option.
- **Dirty** mode: No checking on trunks which create violations with existing routes. Specify this mode with the **-do_not_check_shorts** option.
- **Clean** mode: In this case, no trunks should violate spacing. Specify this mode with the **-avoid_obstruction** option.

The command supports the following track snapping modes:

- Tracks: Trunks should always snap to tracks without overlapping, if an overlap would occur, snap to the next available track by spreading the wires.
- Trackless: Tracks are ignored and routes are created within the minimum spacing distance. Specify this mode with the **-do_not_snap_to_track** option.

The trunk routes derive width from underlying tracks or use the default width if **-do_not_snap_to_track** is specified. If the net has a non-default routing rule (NDR) that requires a width on the layer, the route width is set according NDR width. The spacing between trunk routes can also be adjusted using NDR spacing. To adjust shielding user can set NDR shielding width and spacing.

The command supports multiple levels of physical hierarchy and can operate on a net within any block of a design and create the

corresponding routes within that block.

The command also supports cross-hierarchy routing for top-down flows. The command traces the net across the hierarchy and uses connections of all traced nets to determine the bounding box for the net for the user-specified topology. All routes created would belong to a single top-level net. To finalize the routing, you must use the **push_down_objects** command to push the top-level routing into blocks the route intersects.

EXAMPLES

The following set of commands create a Z-topology connecting pins using layers M2 and M3.

```
prompt> create_topology_edge -name E12 -plan TP1 -nodes {TP1/N1 TP1/N2} \
      -shape_layers {{METAL3 50% 0} {METAL2 100% 0} {METAL3 50% 0}}
prompt> create_trunk_topology -edge TP1/E12
```

SEE ALSO

create_trunk_pin_to_pin(2)
create_trunk_pin_to_trunk(2)
create_topology_plan(2)
create_topology_node(2)
create_topology_edge(2)
push_down_objects(2)

create_tsv_array

Creates an array of through silicon vias (TSVs) in the current design.

SYNTAX

int **create_tsv_array**
-via_def *TSV_via_def_name*
[-name *array_name*]
[-bbox { {*llx lly*} {*urx ury*} }]
[-boundary *polyrect*]
-delta {*dx dy*}
[-pattern inline | staggered_1 | staggered_2]
[-repeat {*columns rows*}]
[-origin {*x y*}]
[-orientation R0 | R90 | R180 | R270 | MY | MXR90 | MX | MYR90]

Data Types

TSV_via_def_name string
array_name string
llx float
lly float
urx float
ury float
polyrect list
dx float
dy float
columns integer
rows integer
x float
y float

ARGUMENTS

-via_def *TSV_via_def_name*

Specifies the TSV via def. This is a required option.

-name *array_name*

Specifies the TSV array name.

-bbox { {*llx lly*} {*urx ury*} }

Specifies the region in which to place the TSV. This option and the **-boundary** option are mutually exclusive. If neither this option

nor **-boundary** option is used, the entire design area is used.

-boundary ***polyrect***

Specifies a *polyrect* region in which to place the TSV. This option and the **-bbox** option are mutually exclusive. If neither this option nor **-bbox** option is used, the entire design area is used. The *polyrect* is described as a list of location coordinates.

-delta {*dx dy*}

Specifies the horizontal pitch (*dx*) between adjacent columns and the vertical pitch (*dy*) between adjacent rows. Min delta value is {cutWidth+minCutSpacing, cutHeight+minCutSpacing}. This is a required option.

-pattern inline | staggered_1 | staggered_2

Specifies the placement pattern of the TSV. Valid values for this option are: inline, staggered_1, and staggered_2. The inline argument places TSV at every grid point. The staggered_1 argument places TSV only at grid points where the sum of column index and row index is even. The staggered_2 argument places TSV only at grid points where the sum of column index and row index is odd. If this option is not specified, the command creates an inline TSV array.

-repeat {*columns rows*}

Specifies the total number of columns and rows in the TSV array. The command does not create TSV outside of the TSV array bounding box, even if you specify a column or row count beyond the bounding box. If this option is not specified, the command fills the bounding box or design area with TSV.

-origin {*x y*}

Specifies the spacing between the lower-left corner of TSV array bounding box and the lower-left corner of the bounding box of the TSV located at column 0 and row 0. If this option is not specified, the command derives the origin so that the TSV array is centered in the bounding box.

-orientation R0 | R90 | R180 | R270 | MY | MXR90 | MX | MYR90

Specifies the orientation of all TSVs in the array. If this option is not specified, the default orientation is R0.

DESCRIPTION

This command creates an array of TSV.

EXAMPLES

The following example creates a TSV array.

prompt> **create_tsv_array -via_def VIAB1 -delta {25 25} -bbox {{10 10} {120 120}}**

SEE ALSO

assign_3d_interchip_nets(2)
assign_tsv(2)

create_undo_marker

Create a named marker that can be used to undo or redo to.

SYNTAX

string **create_undo_marker**
marker_name

Data Types

marker_name string

ARGUMENTS

marker_name

The user name for the new marker.

DESCRIPTION

This command creates a user-named undo marker. Undo markers identify points in the command execution history that can be undone or redone to. It is an error to specify a name for which a marker already exists.

This command creates a user marker. Markers are also created automatically by the undo system whenever an undoable command is executed. Both user and system markers get an automatically-generated system name (distinct from *marker_name*) of the form *SNPS_MARKER_#*. It is an error to specify a user name that has the same form as system names.

Note that markers may automatically become invalid if the undo history is cleared or truncated. The undo history is cleared whenever a non-undoable command is executed, or if the undo system has been disabled via the *undo.enabled* app option. The undo history is automatically truncated based on the size limits imposed by the *undo.max_levels* and *undo.max_memory* app options, and may be truncated in the redo direction if a state-changing command is executed while not at the latest point in the command history.

EXAMPLES

The following creates a marker named *my_marker*:

```
prompt> create_undo_marker my_marker
my_marker
```

SEE ALSO

undo(2)
redo(2)
eval_with_undo(2)
get_undo_info(2)

create_upf2hdl_vct

Defines a value conversion table that can be used to convert UPF supply_net_type values into HDL logic values. This command is supported only in UPF mode.

SYNTAX

status **create_upfhdl_vct**
name
-hdl_type {<vhdl | sv> [*typename*]}
-table {{*from_value to_value*}*}

Data Types

vct_name string
typename string
from_value UPF state type
to_value HDL logic

ARGUMENTS

vct_name

The value conversion table name.

-hdl_type *{<vhdl|sv> typename}*

The HDL type for which the value conversions are defined.

-table *{{<from_value> <to_value>}*}*

A list of UPF state type values to map to the values of the HDL type.

DESCRIPTION

The *create_upf2hdl_vct* command defines a value conversion table for *supply_net_type.state* value when that value is propagated from a UPF supply net into a logic port defined in an HDL. It provides a 1:1 conversion for each possible combination of the partially on and on/off states. *create_upf2hdl_vct* does not check that the values are compatible with any HDL port type.

vct_name provides a name for the value conversion table for later use with the *connect_supply_net* command.

-hdl_type specifies the HDL type for which the value conversions are defined. This information allows a tool to provide completeness and compatibility checks.If the typename is not one of the language's predefined types or one of the types specified in

the next paragraph, then it shall be of the form library.pkg.type.

The following HDL types shall be the minimum set of types supported:

a) VHDL

1) Bit, std_[u]logic, Boolean

2) Subtypes of std_[u]logic

b) SystemVerilog reg/wire, Bit, Logic

-table defines the 1:1 conversions from UPF supply net states to an HDL logic value. The values shall be consistent with the HDL type values.

For example:

- When converting to SystemVerilog logic type, the set of legal values is 0, 1, X, and Z.

- When converting to SystemVerilog or VHDL bit, the legal values are 0 or 1.

- When converting to VHDL std_[u]logic, the legal values are U, X, 0, 1, Z, W, L, H, and -.

The conversion values have no semantic meaning in UPF. The meaning of the conversion value is relevant to the power-aware HDL model to which the supply net is connected.

SEE ALSO

create_hdl2upf_vct(2)
connect_supply_net(2)

create_utilization_configuration

Creates a new utilization configuration.

SYNTAX

create_utilization_configuration
[-capacity *capacity_type*]
[-include *object_type*]
[-exclude *object_type*]
[-scope *config_scope*]
[-as_user_default]
[-force]
config_name

Data Types

capacity_type	string
object_type	string
config_scope	string
config_name	string

ARGUMENTS

-capacity *capacity_type*

Utilization to be based on the given type. The supported types are:

core_area
The core area of the given block will be considered as capacity.

boundary
The boundary of the given block will be considered as capacity.

site_array
If a default site-array is defined for the given block that will be considered as capacity. In the absence of a default site array, the bottom-most site-array will be used as capacity.

site_row
The aggregate area of all the site-rows defined in the block will be considered as capacity. The site-rows belonging to site-arrays will also be considered with this option. This is the default capacity type.

-include *object_type*

Types of objects to be included in utilization computation. The inclusion and exclusion object_type categories must be distinct. Any object_type specified with the -include option can not be repeated with the -exclude option of the same command. The supported types are:

hard_macros
macro_keepouts
soft_macros
io_cells
fixed_cells
physical_only_cells
hard_blockages
soft_blockages
pg_straps
spare_cells
flip_chip_bump_cells
cell_spacing
all (all of the above)
none (none of the above)

-exclude *object_type*

Types of objects to be excluded from utilization computation. The exclusion and inclusion object_type categories must be distinct. Any object_type specified with the -exclude option can not be repeated with the -include option of the same command. The supported types are:

hard_macros
macro_keepouts
soft_macros
io_cells
fixed_cells
physical_only_cells
hard_blockages
soft_blockages
pg_straps
spare_cells
flip_chip_bump_cells
cell_spacing
all (all of the above)
none (none of the above)

-scope *config_scope*

Determines the place where the configuration needs to be stored, thereby controlling the scope of the configuration.

The allowed values for the library-manager shell are:

tech
The tech assocaited with the current library in the workspace. The configurations stored in the tech can be accessed from all the libraries, the tech will be associated with. This is also the default value for this option.

lib
The current library in the workspace.

The allowed values for the design shell are:

tech
The tech associated with the current design-library. The configurations

stored in the tech, can be accessed from all the libraries, the tech will be associated with.

lib
The current design-library. The configurations stored in the library can be accessed from all the designs in this library. This is also the default value for this option.

block
The current block.

-as_user_default

Creates user defined default configuration for utilization reporting.

-force

Force overwrite the existing configuration.

config_name

Specify the name of the configuration

DESCRIPTION

This command can be used to create configurations which control the way the utilization will be calculated/reported. The default configuration has capacity type based on block's core-area and no exclusions.

The configurations created with this command can be passed to the utilization reporting command **report_utilization**

If a configuration with same name or capacity and exclusion values already exists, then the command stops reporting a warning. However, the '-force' option can be used to overwrite the existing configurations.

EXAMPLES

The following example creates a configuration by name config1 with default values and stores it in the current design-library.

```
prompt> create_utilization_configuration config1
```

The following example creates a configuration, in the current block, by name config2 for which the utilization of a block, with respect to its default site-array, will be reported by excluding hard and soft placement blockages.

```
prompt> create_utilization_configuration config2 -scope block \
  -capacity site_array -exclude {hard_blockages soft_blockages}
```

SEE ALSO

get_utilization_configurations(2)
remove_utilization_configurations(2)

report_utilization(2)

create_via

Creates a via in the current design.

SYNTAX

collection **create_via**
-via_def *definition_name*
-origin {*x y*}
[-orientation *orientation*]
[-pitch {*horizontal vertical*}]
[-size {*rows columns*}]
[-cut_pattern *cut_pattern*]
[-shape_use *use*]
[-net *net*]
[-port *port*]
[-lower_mask_constraint *lower_mask*]
[-cut_mask_constraint *cut_mask*]
[-upper_mask_constraint *upper_mask*]

Data Types

definition_name string
x float
y float
orientation string
horizontal float
vertical float
rows integer
columns integer
cut_pattern string
use string
net collection
port collection
lower_mask string
cut_mask string
upper_mask string

ARGUMENTS

-via_def *definition_name*

Specifies the simple or custom via definition from which to create a via.

-origin {*x y*}

Specifies the placement location of the via's origin (i.e., its center).

-orientation *orientation*

Specifies the orientation of the via. Valid values are MX, MXR90, MY, MYR90, R0, R180, R270, and R90. Default orientation is R0.

-pitch {*horizontal vertical*}

Specifies the horizontal and vertical pitch values between cuts of a simple array via. The values must be non-negative. When not specified, the default pitch values are obtained from the via definition.

-size {*rows columns*}

Specifies the number of rows and columns (i.e., cut array size) in a simple array via. The values must be 1 or greater. When not specified, the default cut array size is 1x1. When the size is 1x1, a simple via is created instead of a simple array via.

-cut_pattern *cut_pattern*

Specifies the cut pattern in a simple array via. If specified, this cut pattern overrides the via_def cut pattern. If not specified, the array via inherits the cut pattern from the via_def. The pattern defines a MxN pattern of cuts. If the pattern is smaller than the array via size, the pattern is replicated left-to-right, bottom-to-top in the array via. Excess bits in the pattern are ignored. The pattern is formatted as a space delimited string of equal length substrings (e.g., "1010 0101 1011" is a 3x4 pattern). Each substring must consist of only 1s and 0s, representing a row in the MxN pattern, starting with the bottom row in the pattern. A 1 means the cut is included. A 0 means the cut is omitted.

-shape_use *use*

Specifies the usage of the via. Valid values are area_fill, core_wire, detail_route, follow_pin, global_route, lib_cell_pin_connect, macro_pin_connect, opc, pg_augmentation, ring, shield_route, stripe, user_route, and zero_skew. Default value is detail_route.

-net *net*

Specifies the net that owns this via. If the net is power or ground, the via rule_type attribute is automatically set to "none". Otherwise it is set to "default".

The *-net* and **-port** options are mutually exclusive. A via may be owned by either a net or port, but not both.

-port *port*

Specifies the port that owns this via. The *-net* and **-port** options are mutually exclusive. A via may be owned by either a net or port, but not both.

-lower_mask_constraint *lower_mask*

Specifies the lower mask constraint of the via. Valid values are no_mask, mask_one, mask_two, mask_three, mask_four, and same_mask. Default lower mask constraint is no_mask for simple vias. For custom vias, the default lower mask constraint is inherited from the via_def.

-cut_mask_constraint *cut_mask*

Specifies the cut mask constraint of the via. Valid values are no_mask, mask_one, mask_two, mask_three, mask_four, mask_five, mask_six, mask_seven, mask_eight, mask_nine, mask_ten, mask_eleven, mask_twelve, mask_thirteen, mask_fourteen, mask_fifteen and same_mask. Default cut mask constraint is no_mask for simple vias. For custom vias, the default cut mask constraint is inherited from the via_def.

-upper_mask_constraint *upper_mask*

Specifies the upper mask constraint of the via. Valid values are no_mask, mask_one, mask_two, mask_three, mask_four, and same_mask. Default upper mask constraint is no_mask for simple vias. For custom vias, the default upper mask constraint is

inherited from the via_def.

DESCRIPTION

This command creates a new via and returns a collection containing the new via. Simple, simple array, and custom vias are supported.

A simple via is defined as a single-cut instance of a simple via definition. To create a simple via, specify a simple via definition and origin. The orientation and use may be optionally specified.

A simple array via is defined as a multi-cut instance of a simple via definition. To create a simple array via, specify a simple via definition, origin, and cut array size greater than 1x1. The orientation, pitch values, and use may be optionally specified.

A custom via is defined as an instance of a custom via definition. To create a custom via, specify a custom via definition and origin. The orientation and use may be optionally specified.

A via may be assigned to either a net or port, but not both at the same time. Once assigned, the net or port owns the via. When a via is assigned to a port, a new terminal is created. The new terminal is owned by the port, owns the via, has the default access direction (i.e., all), and has the default auto-generated name.

EXAMPLES

The following example creates a simple via on a net.

```
prompt> create_via -via_def VIA12 -origin {1000.100 200.320} -net Clk
```

The following example creates a simple array via. Its cut pattern is inherited from the via_def.

```
prompt> create_via -via_def VIA12 -origin {1000.100 200.320}
-size {10 10} -pitch {100 200}]
```

The following example creates a custom via on a port.

```
prompt> create_via -via_def CUSTOM_FATVIA12 -origin {3000.100 400.000}
-port Clk
```

The following example creates a 5x5 simple array via with a 2x2 cut pattern. The resulting array via has an alternating cut pattern, in which the lower-left cut is included.

```
prompt> create_via -via_def VIA12 -origin {100 200} -size {5 5}
-pitch {100 200} -cut_pattern "10 01"]
```

The following example creates a 5x5 simple array via with a 5x5 cut pattern, in which the 4 cuts in the upper-right corner of the via is omitted.

```
prompt> create_via -via_def VIA12 -origin {100 200} -size {5 5}
-pitch {100 200} -cut_pattern "11111 11111 11111 11100 11100"]
```

SEE ALSO

get_vias(2)
remove_vias(2)

create_via_def

Creates a via_def in a block or technology file.

SYNTAX

collection **create_via_def**
via_def_name *definition_name*
[-design *design*]
[-library *library*]
[-tech *tech*]
[-force]
[-shapes *shape_list*]
[-lower_layer *layer*]
[-cut_layer *layer*]
[-upper_layer *layer*]
[-cut_size {*horizontal vertical*}]
[-lower_enclosure {*horizontal vertical*}]
[-upper_enclosure {*horizontal vertical*}]
[-min_rows *rows*]
[-min_columns *columns*]
[-min_cut_spacing *spacing*]
[-tsv_keepout_spacing *spacing*]
[-cut_pattern *cut_pattern*]
[-is_default]
[-source_type *contact_source_type*]
[-is_excluded_for_signal_route]
[-mask_pattern *mask_pattern*]
[-upper_mask_pattern *upper_mask_pattern*]
[-lower_mask_pattern *lower_mask_pattern*]
[-redundant_via_insertion_only]

Data Types

definition_name string
design collection
library collection
tech collection
shape_list collection
layer collection
horizontal float
vertical float
rows integer
columns integer
spacing float
cut_pattern string
contact_source_type string
mask_pattern string

upper_mask_pattern string
lower_mask_pattern string

ARGUMENTS

via_def_name *definition_name*

Specifies the name of the simple or custom via_def to create.

-design *design*

Specifies the block in which to create the via_def. If neither a design nor technology file is specified, the via_def is written to the current block.

-library *library*

Specifies the technology library in which to create the via_def. The technology library contained by or referenced by the given library is used. If neither a design nor technology file is specified, the current block is used.

-tech *tech*

Specifies the technology file in which to create the via_def. If neither a design nor technology file is specified, the current block is used.

-force

Forces replacement of an existing via_def of the same name. By default, attempting to create a via_def with an existing name is an error.

-shapes *shape_list*

Specifies a collection of shapes from which to create a custom via_def. The mask constraint is optional and mutual exclusive with the options mask_pattern, -upper_mask_pattern and lower_mask_pattern. The valid values for cut shape are no_mask, mask_one, mask_two, mask_three, mask_four, mask_five, mask_six, mask_seven, mask_eight, mask_nine, mask_ten, mask_eleven, mask_twelve, mask_thirteen, mask_fourteen, mask_fifteen and same_mask, and the default value is no_mask. The valid values for metal shape are no_mask, mask_one, mask_two, mask_three, mask_four, and same_mask. Default mask constraint is no_mask.This option is mutually exclusive with **-cut_layer** and the other options which parameterize a simple via_def.

-lower_layer *layer*

Specifies the lower layer of a via_def.

This argument is required if creating a through-silicon via ("TSV") def. If specified for other via_def types, it must match what would be automatically determined given the specified cut layer and other arguments. Note that this command creates a through-silicon via_def if and only if its *-tsv_keepout_spacing* argument is specified.

-cut_layer *layer*

Specifies the cut layer of a simple via_def.

-upper_layer *layer*

Specifies the upper layer of a via_def.

This argument is required if creating a through-silicon via ("TSV") def. If specified for other via_def types, it must match what would be automatically determined given the specified cut layer and other arguments. Note that this command creates a through-

silicon via_def if and only if its *-tsv_keepout_spacing* argument is specified.

-cut_size {*horizontal vertical*}

Specifies the cut size of a simple via_def.

-lower_enclosure {*horizontal vertical*}

Specifies the lower enclosure size of a simple via_def.

-upper_enclosure {*horizontal vertical*}

Specifies the upper enclosure size of a simple via_def.

-min_rows *rows*

Specifies the minimum number of rows allowed in an array simple via created with this via_def.

-min_columns *columns*

Specifies the minimum number of columns allowed in an array simple via created with this via_def.

-min_cut_spacing *spacing*

Specifies the minimum cut spacing allowed in an array simple via created with this via_def.

-tsv_keepout_spacing *spacing*

Specifies the TSV keepout spacing allowed in a through-silicon via ("TSV") cell created with this via_def.

Note that this command creates a via_def of type "through_silicon_via_def" if and only if this argument is specified.

-cut_pattern *cut_pattern*

Specifies the default cut pattern on an array via created from this simple via_def. If not specified, the array via has a full array of cuts (i.e., no cuts are omitted). If specified, the pattern defines a MxN pattern of cuts. If the pattern is smaller than the array via size, the pattern is replicated left-to-right, bottom-to-top in the array via. Excess bits in the pattern are ignored. The pattern is formatted as a space delimited string of equal length substrings (e.g., "1010 0101 1011" is a 3x4 pattern). Each substring must consist of only 1s and 0s, representing a row in the MxN pattern, starting with the bottom row in the pattern. A 1 means the cut is included. A 0 means the cut is omitted.

-is_default

Specifies whether this via_def is a default via_def for the given metal layers. Only a simple via_def can be a default via_def.

-source_type *contact_source_type*

Specifies the contact source type for this via_def. Allowed values are generated, generated_for_special_nets, multiple_cut, multiple_cut_fixed, single_cut and single_cut_fixed. The default is single_cut_fixed.

-is_excluded_for_signal_route

Specifies whether this via_def can be used by the router for signal route vias or whether it is restricted for use for power vias.

-mask_pattern *mask_pattern*

Specifies the cut layer mask pattern of via_def in a multiple-patterning technology. Valid values are alternate_columns, alternate_rows, alternating and uniform. The default is uniform, which means adjacent cut shapes are assigned the same color; alternating means that adjacent cut shapes are assigned successive colors; alternate_columns means the cut shapes in alternating columns are assigned same colors and the colors of cut shapes in successive columns are successive; alternate_rows means the cut shapes in alternating rows are assigned same colors and the colors of cut shapes in successive columns are successive.

-upper_mask_pattern ***upper_mask_pattern***

Specifies the upper layer mask pattern of via_def in a multiple-patterning technology. Valid values are alternating and uniform. Default is uniform.

-lower_mask_pattern ***lower_mask_pattern***

Specifies the lower layer mask pattern of via_def in a multiple-patterning technology. Valid values are alternating and uniform. Default is uniform.

-is_excluded_for_signal_route

Specifies whether this via_def can only be used by the router for redundant via insertion.

DESCRIPTION

This command creates a new via_def in either block or a tech and returns a collection containing the new via_def. Simple, simple array and custom via_defs are supported.

A simple via_def defines a single_cut simple via. It is specified by a cut layer and the height and width of the rectangular shapes on its cut and metal layers. The metal layers are inferred from the cut layer using the mask order of the layers in the technology.

A simple array via_def defines a multiple-cut simple via. To create a simple array via_def, one or more of min_columns, min_rows or min_cut_spacing must be specified.

A custom via_def defines a custom via from an arbitrary collection of Manhattan polygons. The cut layers, upper metal layer, lower metal layer and the shape mask constraint of the via_def are inferred from the given collection of shapes.

EXAMPLES

The following example creates a simple via_def in the current block with an alternating cut pattern, in which the lower-left cut is included.

```
prompt> create_via_def VIA12_FAT -cut_layer VIA1 -cut_size {100 100} \
  -upper_enclosure {200 200} -lower_enclosure {200 200} \
  -cut_pattern "10" -is_default
```

The following example creates a simple array via_def with a full cut array.

```
prompt> create_via_def VIA12_FAT -cut_layer VIA1 -cut_size {100 100} \
  -upper_enclosure {200 200} -lower_enclosure {200 200} -is_default \
  -min_rows 2 -min_columns 2 -min_cut_spacing 0.2
```

The following example creates a custom via_def.

```
prompt> create_via_def CUSTOM1 -shapes { {M1 {0 0} {0 1} {1 1} {1 0}} \
  {VIA1 {0.5 0.5} {0.7 0.7}} {M2 {0 0} {1 1}} }
```

The following example creates a custom via_def with shape mask constraints.

```
prompt> create_via_def CUSTOM1 -shapes { {M1 {0 0} {0 1} {1 1} {1 0} mask_one} \
  {VIA1 {0.5 0.5} {0.7 0.7} mask_two} {M2 {0 0} {1 1} mask_two} }
```

SEE ALSO

get_via_defs(2)
report_via_defs(2)

create_via_ladder

Creates a via ladder in the current design.

SYNTAX

collection **create_via_ladder**
-shapes *shape_list*
[-via_rule *via_rule_name*]
[-electromigration *boolean_option*]
[-pattern_must_join *boolean_option*]
[-high_performance *boolean_option*]
[-pin *pin*]

Data Types

shape_list collection
via_rule_name string
boolean_option boolean
pin string

ARGUMENTS

-shapes *shape_list*

Specifies a list of shapes to be added to the via ladder object. The list may contain shape collections, specified using the **get_shapes** command. All the shapes (vias and layer shapes) must be connected to the same net. This is a required option for this command.

-via_rule *via_rule_name*

A string value which represents the via rule name for the created via ladder. It has a default value of empty string.

-electromigration *boolean_option*

A boolean value which represents the type of via ladder. If this is set to true, then the via ladder is of type 'Electromigration'. It has a default value false.

-high_performance *boolean_option*

A boolean value which represents the type of via ladder. If this is set to true, then the via ladder is of type 'High Performance'. It has a default value false.

-pattern_must_join *boolean_option*

A boolean value which represents the type of via ladder. If this is set to true, then the via ladder is of type 'Pattern Must Join'. It has

a default value false.

-pin ***pin***

A string value which represents the pin for the created via ladder. It should be a physical pin.

DESCRIPTION

This command creates a new via ladder object and returns a collection containing the new via ladder.

EXAMPLES

The following example creates a via ladder with list of shapes.

```
prompt> create_via_ladder -shapes {PATH_31_1 VIA_S_70} \
  -via_rule rule_1 -electromigration -pin icc_route2/VDD
{VIA_LADDER_0}
```

The following example creates a via ladder with collection of shapes returned by get_shapes command.

```
prompt> create_via_ladder -shapes [get_shapes *] \
  -via_rule rule_2 -high_performance -pattern_must_join
{VIA_LADDER_1}
```

SEE ALSO

get_via_ladders(2)
remove_via_ladders(2)

create_via_matrix

Creates a via matrix in the current design.

SYNTAX

collection **create_via_matrix**
-via_def *definition_name*
-origin {*x y*}
-pitch {*horizontal vertical*}
-size {*rows columns*}
-net {*net*}
[-base_pattern *base_pattern*]
[-lower_mask_pattern *lower_mask_pattern*]
[-cut_mask_pattern *cut_mask_pattern*]
[-upper_mask_pattern *upper_mask_pattern*]
[-orientation *orientation*]
[-via_orientation *orientation*]
[-pattern *via_pattern*]
[-shape_use *use*]
[-lower_mask_constraint *lower_mask*]
[-cut_mask_constraint *cut_mask*]
[-upper_mask_constraint *upper_mask*]
[-base_pitch {*base_horizontal base_vertical*}]
[-base_size {*base_rows base_columns*}]

Data Types

definition_name string
x float
y float
horizontal float
vertical float
rows integer
columns integer
base_pattern string
lower_mask_pattern string
cut_mask_pattern string
upper_mask_pattern string
orientation string
via_pattern string
use string
net collection
lower_mask string
cut_mask string
upper_mask string
base_horizontal float

base_vertical float
base_rows integer
base_columns integer

ARGUMENTS

-via_def *definition_name*

Specifies the simple or custom via definition from which to create a via matrix.

-origin {*x y*}

Specifies the placement location of the via matrix's origin (i.e., center of lower-left base via).

-pitch {*horizontal vertical*}

Specifies the horizontal and vertical pitch values between base vias of a via matrix.

-size {*rows columns*}

Specifies the number of rows and columns (i.e., base via array size) in a via matrix. The values must be 1 or greater and at least one of them is greater than 1.

-base_pattern *base_pattern*

Specifies the cut pattern in a base via of the via matrix. If specified, this cut pattern overrides the via_def cut pattern. If not specified, the array via inherits the cut pattern from the via_def. The pattern defines a MxN pattern of cuts. If the pattern is smaller than the array via size, the pattern is replicated left-to-right, bottom-to-top in the base array via. Excess bits in the pattern are ignored. The pattern is formatted as a space delimited string of equal length substrings (e.g., "1010 0101 1011" is a 3x4 pattern). Each substring must consist of only 1s and 0s, representing a row in the MxN pattern, starting with the bottom row in the pattern. A 1 means the cut is included. A 0 means the cut is omitted.

-lower_mask_pattern *lower_mask_pattern*

Specifies the lower layer mask pattern of base vias in the via_matrix in a multiple-patterning technology. Valid values are arbitrary, uniform, alternate_columns and alternate_rows. Default is uniform.

-cut_mask_pattern *cut_mask_pattern*

Specifies the cut layer mask pattern among all base vias in a multiple-patterning technology. Valid values are arbitrary, uniform, alternate_columns and alternate_rows. Default is uniform. which means cut mask values in all base vias of the via matrix are the same. When it is arbitrary, cut masks of the lower left cuts in the base vias are considered independently. The cut mask pattern in a base via is inherited from the via_def of the via matrix.

-upper_mask_pattern *upper_mask_pattern*

Specifies the upper layer mask pattern of base vias in the via_matrix in a multiple-patterning technology. Valid values are arbitrary, uniform, alternate_columns and alternate_rows. Default is uniform.

-pattern *via_pattern*

Specifies the via pattern in a via matrix. The pattern defines a MxN pattern of base vias. If the pattern is smaller than the via matrix size, the pattern is replicated left-to-right, bottom-to-top in the via matrix. Excess bits in the pattern are ignored. The pattern is formatted as a space delimited string of equal length substrings (e.g., "1010 0101 1011" is a 3x4 pattern). Each substring must consist of only 1s and 0s, representing a row in the MxN pattern, starting with the bottom row in the pattern. A 1 means the base via is included. A 0 means the base via is omitted. If not specified, all the base vias are included.

-orientation *orientation*

Specifies the orientation of the via matrix. Valid values are MX, MXR90, MY, MYR90, R0, R180, R270, and R90. Default orientation is R0.

-via_orientation *orientation*

Specifies the orientation of base vias in the via matrix. Valid values are MX, MXR90, MY, MYR90, R0, R180, R270, and R90. Default orientation is R0.

-shape_use *use*

Specifies the usage of the via matrix. Valid values are area_fill, core_wire, detail_route, follow_pin, global_route, lib_cell_pin_connect, macro_pin_connect, opc, ring, shield_route, stripe, user_route, and zero_skew. Default value is stripe.

-net *net*

Specifies the power/ground net that owns this via matrix.

-lower_mask_constraint *lower_mask*

Specifies the lower mask constraint of the via matrix. Valid values are no_mask, mask_one, mask_two, mask_three, mask_four, and same_mask. Default lower mask constraint is no_mask. When the mask_pattern is alternate_rows or alternate_columns, and uniform, only the mask of lower-left base via needs to be specified. For alternate_rows or alternate_columns, another mask is specified according to the lower_layer of via_def number_of_masks attribute. If the number_of_masks is zero, alternate_rows or alternate_columns works as in uniform style.

-cut_mask_constraint *cut_mask*

Specifies the cut mask constraint of the via matrix. Valid values are no_mask, mask_one, mask_two, mask_three, mask_four, mask_five, mask_six, mask_seven, mask_eight, mask_nine, mask_ten, mask_eleven, mask_twelve, mask_thirteen, mask_fourteen, mask_fifteen and same_mask. Default cut mask constraint is no_mask. When the mask_pattern is alternate_rows or alternate_columns, uniform, only the mask of lower-left base via needs to be specified. For alternate_rows or alternate_columns, another mask is specified according to the cut_layer of via_def number_of_masks attribute. If the number_of_masks is zero, alternate_rows or alternate_columns works as in uniform style.

-upper_mask_constraint *upper_mask*

Specifies the upper mask constraint of the via matrix. Valid values are no_mask, mask_one, mask_two, mask_three, mask_four, and same_mask. Default upper mask constraint is no_mask. When the mask_pattern is alternate_rows or alternate_columns, and uniform, only the mask of lower-left base via needs to be specified. For alternate_rows or alternate_columns, another mask is specified according to the upper_layer of via_def number_of_masks attribute. If the number_of_masks is zero, alternate_rows or alternate_columns works as in uniform style.

-base_pitch {*horizontal vertical*}

Specifies the horizontal and vertical pitch values between cuts in the base vias of a via matrix. The values must be non-negative. When not specified, the default base pitch values are got from the via definition.

-base_size {*rows columns*}

Specifies the number of rows and columns (i.e., cut array size) in a base simple array via of the via matrix. The values must be 1 or greater. When not specified, the default cut array size is 1x1. When the base size is 1x1, a base simple via is created instead of a base simple array via.

DESCRIPTION

This command creates a new via matrix and returns a collection containing the new via matrix. The base via of the via matrix can be one of a simple, simple array, or custom via.

A simple base via is defined as a single-cut instance of a simple via definition. To create a simple base via, specify a simple via definition and origin. The orientation and use may be optionally specified.

A simple array base via is defined as a multi-cut instance of a simple via definition. To create a simple array base via, specify a simple via definition, origin, and cut array size greater than 1x1. The orientation, pitch values, and use may be optionally specified.

A custom base via is defined as an instance of a custom via definition. To create a base custom via, specify a custom via definition and origin. The orientation and use may be optionally specified.

A via matrix may be assigned to either a net. Once assigned, the net owns the via matrix.

Note that the default value for the -shape_use option is "detail_route". This is probably not what you want. The placer will not check for legal DRC in relation to detail_route shapes. The placer's assumption is that the detail route will be moved later. For power vias, the shape_use should be set to "stripe". For clock meshes, you can use "user_route". In both of these cases, the placer will be sure to place cells so that they don't have bad DRC interactions with your via matrix.

EXAMPLES

The following example creates a via matrix with a base simple via on a net.

```
prompt> create_via_matrix -via_def VIA12 -origin {1000.100 200.320} \
  -pitch {10 10} -size {2 3} -net Clk -shape_use user_route
```

The following example creates a via matrix with a base simple array via. Its base pattern is inherited from the via definition.

```
prompt> create_via_matrix -via_def VIA12 -origin {1000.100 200.320} \
  -pitch {10 10} -size {2 3} -base_size {3 4} -net clear
```

The following example creates a via matrix with a base custom via.

```
prompt> create_via_matrix -via_def FATVIA12_230_530_ALL_1_2 \
  -pitch {10 10} -size {20 30} -origin {1000.100 200.320} -net clear
```

The following example creates a 5x5 via matrix with a base simple via with a 2x2 via pattern. The resulting via matrix has an alternating via pattern, in which the lower-left cut via is included.

```
prompt> create_via_matrix -via_def VIA12 -origin {100 200} \
  -size {5 5} -pitch {100 200} -pattern "10 01"
```

The following example creates a 5x5 via matrix with a 5x5 pattern, in which the 4 base vias in the upper-right corner of the via matrix are omitted.

```
prompt> create_via_matrix -via_def VIA12 -origin {100 200} \
  -size {5 5} -pitch {100 200} -pattern "11111 11111 11111 11100 11100" \
  -net VDD -shape_use stripe
```

SEE ALSO

get_via_matrixes(2)
remove_via_matrixes(2)

create_via_region

Creates a new via_region in frame block

SYNTAX

collection **create_via_region**
-terminal *terminal*
-via_def *via_defs*
-boundary *coord_list*
[-design *design*]
[-rotate]
[-force]

Data Types

terminal	collection
via_defs	collection
coord_list	collection
design	collection

ARGUMENTS

-terminal *terminal*

Specifies the terminal collection for which to create the via region. The boundary of the via region must be enclosed by the bounding box of the terminal. Otherwise, the command issues an error message.

-via_def *via_defs*

Specifies the via definition of the via that can be used to make connections to the terminal associated with via region.

-boundary *coord_list*

Specifies the bounding box in the form of coordination list {{x0 y0} {x1 y1} {x2 y2} ...}. The points must specify a rectilinear polygon.

-design *design*

Specifies the frame block in which to create the new via_region. By default, the command creates the new via region in current frame block.

-rotate

Specifies whether the router must rotate the vias by 90-degrees for placement in the via region.

-force

Specifies whether the existing via regions for the pin will be removed before creating the new via region. By default, the existing via regions for the pin will not be removed.

DESCRIPTION

This command creates a new via_region for an existing terminal object, and returns a collection containing the new via_region or empty collection if any error.

The via_region should specify the shape and associated a via_def object, and the via_def should be defined in the design or technology file.

EXAMPLES

The following example creates a via_region in the current frame block.

```
prompt> create_via_region -terminal [get_pins "IN"] -via_def VIA12_FAT \
  -boundary {{100 200} {700 900}}
```

SEE ALSO

get_via_regions(2)
report_via_regions(2)

create_via_rule

Creates a via_rule in a block or technology file.

SYNTAX

collection **create_via_rule**
-name *via_rule_name*
[-design *design*]
[-library *library*]
[-tech *tech*]
[-cut_layer_names {*name_list*}]
[-cut_names {*name_list*}]
[-cut_rows {*int_list*}]
[-cuts_per_row {*int_list*}]
[-em_factor *float*]
[-max_num_stagger_tracks_for_top_level_ndr_track_use *int*]

Data Types

via_rule_name string or collection
design collection
library collection
tech collection
name_list list
int_list list of integers
float float
int interger

ARGUMENTS

-name *via_rule_name*

Specifies the name of the via_rule to be created.

-design *design*

Specifies the block in which to create the via_rule. If neither a design nor technology file is specified, the via_rule is created in the current block.

-library *library*

Specifies the technology library in which to create the via_rule. The technology library contained by or referenced by the given library is used. If neither a design nor technology file is specified, the current block is used.

-tech *tech_object*

Specifies the technology in which to create the via rule. The default is the technology for the current library.

-cut_layer_names *name_list*

Defines the cut layer's names. This is must for defining a via_ladder. If specified then other via_ladder related attributes are must and length of this list attribute should be same as of other via_ladder related attributes.

-cut_names *name_list*

Defines the cut table names. This is must for defining a via_ladder. If specified then other via_ladder related attributes are must and length of this list attribute should be same as of other via_ladder related attributes.

-cut_rows *int_list*

Defines the number of cut rows. This is must for defining a via_ladder. If specified then other via_ladder related attributes are must and length of this list attribute should be same as of other via_ladder related attributes.

-cuts_per_row *int_list*

Defines the number of cuts per row. This is must for defining a via_ladder. If specified then other via_ladder related attributes are must and length of this list attribute should be same for other via_ladder related attributes.

-em_factor *float*

Defines the EM factor of via_ladder rule. This is float type and its value must be positive. This is an optional attribute and may be specified only for via_ladder. If specified then other via_ladder related attributes are mandatory.

-max_num_stagger_tracks_for_top_level_ndr_track_use *int*

Defines the maximum number of stagger tracks for use of top level tracks of via_ladder rule. This is an integer type and its value must be positive. This is an optional attribute and may be specified only for via_ladder. If specified then other via_ladder related attributes are mandatory.

RETURN VALUE

The command returns the created via_rule (as a collection), an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

DESCRIPTION

This command creates a new via_rule in the specified block or tech.

EXAMPLES

The following example creates a via_rule in the current block.

```
prompt> create_via_rule -name VR1
```

The following example creates a via_rule in the tech of current lib.

```
prompt> create_via_rule -library [current_lib] -name VR1
```

The following example creates a via_ladder type of via_rule.

```
prompt> create_via_rule -name VL1 -cut_layer_names {VIA1 VIA2} \
  -cut_names {cut1 cut2} -cut_rows {2 2} -cuts_per_row {2 2}
```

SEE ALSO

get_via_rules(2)
remove_via_rules(2)
report_via_rules(2)

create_virtual_connection

Creates the virtual connection to the specified pins or ports.

SYNTAX

status **create_virtual_connection**
-name *name*
-pins *pin_port_list*
[-weight *weight*]

Data Types

name	string
pin_port_list	collection
weight	integer

ARGUMENTS

-name *name*

Specifies the name of the virtual connection. The command errors out if the specified virtual connection exists.

-pins *pin_port_list*

Specifies a list of pins and ports to be connected to the newly created virtual connection. The command errors out if any of given pins or ports has been connected to other virtual connection already.

-weight *weight*

Specifies the virtual connection weight. The value is integer type and should be great than or equal to 1. If the option is not used, the default weight for virtual connection is 1.

The net weight represents the spring force between eco cells. The bigger the net weight, the higher the force. Currently the command **place_eco_cells** uses connectivity based coarse placement.

By default, all net weight are same. So the eco cell is placed with even distance to driver and load. You can decide how close the eco cells by changing the net weight.

DESCRIPTION

This command creates the virtual connection to the specified pins or ports. It is used before *place_eco_cells -use_virtual_connection*, that performs placement based on user-defined virtual connection. Please refer to the man page of

place_eco_cells for the flow usage about virtual connection based coarse placement.

User should make sure the virtual connection is valid. That is, the virtual connection has one driver and one load at least. The command place_eco_cells will ignore the invalid connection for eco cell automaticlly during coarse placement.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates the virtual connection.

```
prompt> create_virtual_connection \
  -name SNPS_VC_0 \
  -pins {U1/Y lh1/ff1/A}

prompt> create_virtual_connection \
  -name SNPS_VC_1 \
  -pins {lh1/ff1/Y lh1/ff2/A}

prompt> create_virtual_connection \
  -name SNPS_VC_2 \
  -pins {lh1/ff2/Y lh1/ff3/A}

prompt> create_virtual_connection \
  -name SNPS_VC_3 \
  -pins {lh1/ff3/Y U2/A U3/A}
```

SEE ALSO

add_pins_to_virtual_connection(2)
get_attribute(2)
get_virtual_connections(2)
place_eco_cells(2)
remove_pins_from_virtual_connection(2)
remove_virtual_connections(2)
set_attribute(2)

create_voltage_area

Creates a voltage area in the current block.

SYNTAX

status **create_voltage_area**
-power_domains *domain_list*
[-power *supply_net*]
[-ground *supply_net*]
[-nwell *supply_net*]
[-pwell *supply_net*]
[-cells *cell_list*]
[-region *region_list*
[-guard_band *width_list*]
[-is_fixed]
[-merge_regions]
[-target_utilization *utilization*]
[-name *voltage_area_name*]
[-cell *cell_name*]

Data Types

domain_list	list
supply_net	string
cell_list	collection
region_list	list
width_list	list
utilization	float
voltage_area_name	string
cell_name	string or collection

ARGUMENTS

-power_domains *domain_list*

Specifies the power domains to associate with this voltage area. All power domains must be associated with the current block and have the same primary supply net.

-power *supply_net*

Specifies the power supply net for voltage area. The net must match one of the available power supply nets on the power domains.

This option is required for a gas-station type voltage area.

-ground *supply_net*

Specifies the ground supply net for the voltage area. The net must match one of the available ground supply nets on the power domains.

This option is required for a gas-station type voltage area.

-nwell *supply_net*

Specifies the nwell supply net for the voltage area. The net must match one of the available power supply nets on the power domains.

-pwell *supply_net*

Specifies the pwell supply net for the voltage area. The net must match one of the available ground supply nets on the power domains.

-cells *cell_list*

Specifies the cells to be included in the voltage area. This option is used to create a gas-station type voltage area. All cells must be in the same physical hierarchy as the current block. Cells can be physical leaf cells or hierarchical cells. When you specify hierarchical cells, the cells are expanded into the set of physical leaf cells that reside under them. If cells are later added or removed from this hierarchical cell, the cell membership of the voltage area is not automatically updated. Note that one hierarchical cell can expand into multiple physical leaf cells. If the hierarchical cell is purely logical (it corresponds to a local module that has no physical representation), the cell maps to an empty set of physical leaf cells, and the tool issues a warning message.

The specified cells cannot already be explicitly or implicitly associated with another voltage area. When used with the **-power_domains** option, the specified cells must also belong to one of the power domains. If any of the power domains already have an associated voltage area with implicit cell membership (whose **use_power_domain_cells** attribute is **true**), the cells are excluded from the implicit cell list. This ensures that each cell is associated with only one voltage area.

-region *region_list*

Specifies the rectangles or rectilinear polygons of the voltage area.

Each region can be one of the following objects:

- Rectangle
  Each rectangle is specified by its lower-left and upper-right coordinates as {*llx lly*} {*urx ury*}).
- Polygon
  Each polygon is specified by its points as {*x1 y1*} {*x2 y2*} {*x3 y3*} {*x4 y4*} and so on.
- Geometric object A geometric object is the combined area of a mixed collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area include the areas of each object. In the case of layers, the resulting area includes the area of every shape on the layer. Overlapping areas are preserved and not merged.

Each region defines a target placement area for the objects in the voltage area. The specified regions can overlap or be disjoint. The coordinates are relative to the chip origin.

This option is required for a gas-station type voltage area.

-guard_band *width_list*

Specifies the horizontal and vertical guard band widths for the voltage area regions.

A guard band is the space along the boundary of a voltage area shape where cells cannot be placed. A guard band expands outward from the region.

If you do not specify this option, the guard band width is zero and the voltage areas do not have guard bands.

If you specify this option, the specified values must be positive floating point numbers. You must specify a pair of horizontal and vertical guard band widths for each region specified in the **-region** option, except when the specified regions abut or overlap and you also specify the **-merge_regions** option. In this case, you must specify the guard bands for each resulting region.

This option can be used only with the **-region** option.

This option is valid only for primary voltage areas. It is not allowed on gas-station type voltage areas because gas-stations cannot have guard bands.

-is_fixed

Specifies whether the voltage area is at a fixed location. This option can be used only with the **-region** option.

-merge_regions

Merges all abutted and overlapping shapes of this voltage area into the shape with the highest stacking order. This option can be used only with the **-region** option.

-target_utilization *utilization*

Specifies the target utilization for this voltage area during shaping. The *utilization* value must be between 0.1 and 1.0.

If you do not specify this option, the tool uses the target utilization for the block, whether that value is valid or not. The target utilization of the voltage area is not automatically updated when the design's target utilization is changed later on.

This option can be used only with the **-region** option.

This option is valid only for primary voltage areas. It is not allowed on gas-station type voltage areas.

-name *voltage_area_name*

Specifies the name of the voltage area. The name cannot conflict with any existing voltage areas in the current block. This option is required unless the **-power_domains** option is used and a single power domain not already associated with a voltage area is specified. In that case, the name of the voltage area is the full hierarchical name of the power domain.

The DEFAULT_VA name is reserved for the automatically generated default voltage area and cannot be used. If you specify a power domain named DEFAULT_VA, the tool issues a warning message and adds a numeric suffix to the name to avoid conflict, for example, DEFAULT_VA_1.

-cell *cell_name*

Specifies the physical cell in which to add the voltage area. The voltage area is created in the cell's reference block using the coordinate system of the cell argument's top block. The cell must reference a block, not a library cell, unless this command is executed in the library manager. In the library manager, voltage areas can be created in library cells. When not specified, the voltage area is created in the current block.

DESCRIPTION

This command creates a voltage area with the specified geometry on the core area of the chip. A voltage area is associated with a power domain. The placer treats the voltage area as an exclusive, hard move bound and tries to place all the cells associated with the voltage area within the voltage area's region and all the cells not associated with the voltage area outside of it.

The voltage area's region contains one or more rectilinear polygons and rectangles. These shapes can overlap to form a single region, or be disjoint to form multiple regions. There is a single unique voltage area at any given location on the chip. Voltage areas can be physically nested. A voltage area region can exist completely inside another voltage area's region. A voltage area region

can also exist partially inside another voltage area's region, creating two abutted voltage areas. The order of creation determines the precedence of the overlapping regions. The voltage area that was created last has the highest precedence. The effective region of the voltage area with higher precedence preserves its original shape while the voltage area with lower precedence gets its effective shape by subtracting the overlapping region. Shape precedence is decided by an explicit stacking order that is initially the same as the order of creation, with the most recently created voltage area shape having the highest precedence. You can adjust the stacking order by using the **set_voltage_area_shape** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a voltage area from the INST power domain at the rectangle with its lower-left corner at 215,215 and its upper-right corner at 350,350. The x- and y-guard band widths are 1 and 2:

```
prompt> create_voltage_area -power_domains INST -region {{215 215} {350 350}} \
  -guard_band {{1 2}}
{INST}
```

The following example creates a voltage area from the INST1 power domain at the polygon with coordinates {100 100}, {300 100}, {300 200}, {200 200}, {200 300}, and {100 300}.

```
prompt> create_voltage_area -power_domains INST1 \
-region {{{100 100} {300 100} {300 200} {200 200} {200 300} {100 300} {100 100}}}
{INST1}
```

SEE ALSO

create_voltage_area_shape(2)
get_voltage_area_shapes(2)
get_voltage_areas(2)
remove_voltage_area_shapes(2)
remove_voltage_areas(2)
report_voltage_areas(2)
set_voltage_area(2)
set_voltage_area_shape(2)

create_voltage_area_rule

Creates rules for non-native net routing across voltage areas in the current block.

SYNTAX

status **create_voltage_area_rule**
[-name *rule_name*]
[-allow_pass_through true | false]
[-allow_buffering true | false]
[-allow_physical_feedthrough true | false]
[-allow_logical_feedthrough true | false]
[-include_logical_feedthrough_hierarchy *cell_list*]
[-exclude_logical_feedthrough_hierarchy *cell_list*]
-default_rule
-voltage_areas *voltage_areas*

Data Types

rule_name string
cell_list collection
voltage_areas collection

ARGUMENTS

-name *rule_name*

Specifies the name of the voltage area rule. This option must be specified if the **-voltage_areas** option is specified. This option cannot be specified if the **-default_rule** option is specified.

-allow_pass_through true | false

Specifies whether nets can pass through the voltage area. After the rule is created, this property can be modified by calling **set_attribute** on the **is_pass_through_allowed** attribute. By default, this option is true.

-allow_buffering true | false

Specifies whether buffering is allowed in the voltage area. After the rule is created, this property can be modified by calling **set_attribute** on the **is_buffering_allowed** attribute. By default, this option is true.

-allow_physical_feedthrough true | false

Specifies whether physical feedthrough buffering is allowed through the voltage areas. If physical feedthrough buffering is allowed, you must also specify **-allow_pass_through true** and **-allow_buffering true**. It is an error to allow physical feedthroughs, but disallow pass throughs and buffering. After the rule is created, this property can be modified by calling **set_attribute** on the **is_physical_feedthrough_allowed** attribute. The default is false.

-allow_logical_feedthrough true | false

Specifies whether logical feedthrough buffering is allowed through the voltage areas. If logical feedthrough buffering is allowed, you must also specify **-allow_pass_through true** and **-allow_buffering true**. It is an error to allow logical feedthroughs, but disallow pass throughs and buffering. After the rule is created, this property can be modified by calling **set_attribute** on the **is_logical_feedthrough_allowed** attribute. The default is false.

-include_logical_feedthrough_hierarchy *cell_list*

Specifies the hierarchies in which the tool may insert logical feedthrough buffers. Hierarchies are specified by listing logical hierarchical cells to the option. Specify the top-level of hierarchy with the "." character. The specified cells and top-level of hierarchy must be roots of the specified voltage area's power domain. If this option is specified, the **-allow_logical_feedthrough true** option value must be specified and exactly one voltage area must be specified through the **-voltage_areas** option.

The **-include_logical_feedthrough_hierarchy** and **-exclude_logical_feedthrough_hierarchy** options are mutually exclusive. Only one may be specified. If neither are specified, then the **-allow_logical_feedthrough** option applies to all hierarchies.

-exclude_logical_feedthrough_hierarchy *cell_list*

Specifies the hierarchies in which the tool may not insert logical feedthrough buffers. Hierarchies are specified by listing logical hierarchical cells to the option. Specify the top-level of hierarchy with the "." character. The specified cells and top-level of hierarchy must be roots of the specified voltage area's power domain. If this option is specified, the **-allow_logical_feedthrough true** option value must be specified and exactly one voltage area must be specified through the **-voltage_areas** option.

The **-include_logical_feedthrough_hierarchy** and **-exclude_logical_feedthrough_hierarchy** options are mutually exclusive. Only one may be specified. If neither are specified, then the **-allow_logical_feedthrough** option applies to all hierarchies.

-default_rule

Creates the default rule with the name "DEFAULT_RULE".

The **-default_rule** and **-voltage_areas** options are mutually exclusive. One of them must be specified.

-voltage_areas *voltage_areas*

Specifies the voltage areas on which to apply the rule in the current block. More than one voltage area can be specified, and the rule will apply to each one. This command can also be called incrementally with new voltage areas on a previously defined rule. In this case, the rule is added to those voltage areas and retained on previous voltage areas. If a voltage area previously had a voltage area rule defined, and a different one is then applied, then the new rule overrides the old rule.

The **-default_rule** and **-voltage_areas** options are mutually exclusive. One of them must be specified.

DESCRIPTION

This command creates a voltage area rule in the current block. The rule can be either the default rule or a voltage area specific rule. The default rule is automatically applied to all voltage area in the block which are not explicitly assigned a rule. A voltage area specific rule applies to a set of one or more voltage areas in the block.

If the **-default_rule** option is specified, the default rule is created and assigned the name "DEFAULT_RULE". If the default rule already exists, this command returns an error and no rule is created or modified.

If the **-voltage_areas** option is specified, a voltage area specific rule is created for the specified voltage_areas. If the specified name, pass through, physical feedthrough, and logical feedthrough values match an existing rule, the existing rule is applied to the specified voltage areas and no new rule is created.

If a voltage_area in the **-voltage_areas** option is already explicitly assigned an existing rule, this new rule overrides the existing rule. If an existing rule is orphaned (no longer constrains any voltage area), it is removed from the block. If a voltage area is removed

from the block which causes a rule to be orphaned, the rule is removed from the block.

Voltage area rules specify whether nonnative net routing is permitted in the voltage areas. The routing behavior (either to allow routing to pass through the voltage area or not) is specified with the **-allow_pass_through** option.

These routing pass through rules do not apply to all nets, only to nets that are nonnative to the voltage area on which the rule is applied. Native routing is unaffected, and can always pass through the native voltage areas. For example, suppose there is a driver in VA1, a load in VA2, and the net connecting them only has segments in the hierarchies belonging to VA1 and VA2. This net is said to be "native" to both VA1 and VA2, and so any voltage area routing rules applied to either VA1 or VA2 do not apply to this net. Now consider an unrelated voltage area VA3 which is physically between VA1 and VA2. It might be desirable to allow this net to pass through VA3 to connect the driver and load, to make a shorter connection. Or, it might be desirable to disallow this pass through routing in VA3 and force the net to detour around, because of congestion or other reasons. A voltage area rule applied on VA3 will control this behavior for this net, because it is nonnative to VA3.

These routing pass through rules are honored by all optimization engines, including both datapath and clock tree synthesis (CTS) buffer tree topology generation, virtual routing, global routing, track routing, and detail routing.

Physical feedthrough (PFT) buffering is the buffering counterpart to the pass through routing concept. PFT buffering is buffering done across voltage areas that are not native to the hierarchy that the buffers logically belong to. PFT buffering is disabled by default, but can be enabled locally by specifying **-allow_physical_feedthrough true** or globally by setting the **opt.common.allow_physical_feedthrough** application option to *true*. PFT buffering can only be controlled globally by this application option. However, buffering cannot be done when routing cannot be done. So, if a voltage area has a rule that disables pass through routing, it also implies that physical feedthrough buffering is disabled for that voltage area.

Logical feedthrough (LFT) buffering is a complementary solution to PFT buffering. While PFT buffering resolves a buffer's mismatched logical and physical hierarchy membership by setting one of the voltage area's power domains directly on the buffer, LFT buffering moves the buffer to a logical hierarchy under one of the voltage area's power domains.

Voltage area rules are data model objects that can be queried with the **get_voltage_area_rules** command. Each voltage area also has a *voltage_area_rules* attribute which stores the defined rules on the voltage area.

Use the **report_voltage_area_rules** command to report the currently applied rules.

Use the **remove_voltage_area_rules** command to remove any defined rules.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a default voltage area rule named "DEFAULT_RULE" that allows pass throughs and regular buffering, but not physical and logical feedthrough buffering.

```
prompt> create_voltage_area_rule -default_rule \
  -allow_pass_through true -allow_buffering true \
  -allow_physical_feedthrough false -allow_physical_feedthrough false
```

The following example creates a voltage area rule named "RULE1" that allows pass throughs, regular buffering, physical feedthrough buffering, and logical feedthrough buffering on voltage area "VA1".

```
prompt> create_voltage_area_rule -name RULE1 \
  -allow_pass_through true -allow_buffering true \
  -allow_physical_feedthrough true -allow_logical_feedthrough true \
  -voltage_areas VA1
```

The following example applies existing voltage area rule "RULE1" to voltage area "VA2". RULE1 still remains applied to VA1.

```
prompt> create_voltage_area_rule -name RULE1 \
  -allow_pass_through true -allow_buffering true \
  -allow_physical_feedthrough true -allow_logical_feedthrough true \
  -voltage_areas VA2
```

The following example creates a voltage area rule named "RULE2" and applies it to voltage area "VA1", overriding its currently assigned rule "RULE1". Now only RULE2 applies to VA1.

```
prompt> create_voltage_area_rule -name RULE2 -allow_pass_through true \
  -allow_physical_feedthrough false -voltage_areas VA1
```

The following example creates a voltage area rule named "RULE7" which allows logical feedthrough buffers in the I_ORCA_TOP/I_SDRAM_IF level of hierarchy.

```
prompt> create_voltage_area_rule -name RULE7 \
  -allow_logical_feedthrough true -voltage_areas PD7 \
  -include_logical_feedthrough_hierarchy {I_ORCA_TOP/I_SDRAM_IF}
```

SEE ALSO

get_voltage_area_rules(2)
remove_voltage_area_rules(2)
report_voltage_area_rules(2)

create_voltage_area_shape

Creates a voltage area shape at the specified region for providing placement constraints for cells associated with the region.

SYNTAX

collection **create_voltage_area_shape**
-voltage_area *voltage_area*
-region { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } |
geometric_objects
[-guard_band {*x_width y_width*}]
[-target_utilization *utilization*]
[-cells *cell_list*]
[-exclusive]
[-merge_regions]
[-top | -bottom | -above *voltage_area_shape* | -below *voltage_area_shape*]

Data Types

voltage_area	string or collection
llx	float
lly	float
urx	float
ury	float
x	float
y	float
geometric_objects	collection
x_width	int
y_width	int
utilization	float
cell_list	string or collection
voltage_area_shape	string or collection

ARGUMENTS

-voltage_area *voltage_area_object*

Specifies the voltage_area in which to create this shape.

-region { {*llx lly*} {*urx ury*} } | {{*x y*} {*x y*} {*x y*} {*x y*}...} | *geometric_objects*

Specifies the boundary of the voltage area shape. The boundary can be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates as {*llx lly*} {*urx ury*}). A polygon is specified by its points as {*x y*} {*x y*} {*x y*} {*x y*} and so on.

A polygon can also be specified as the combined area of a mixed collection of objects with physical geometry, such as

poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

When specifying the boundary polygon as a collection of physical objects, the resulting area must resolve to a single, connected polygon. It is an error to specify a collection of objects with a combined area that resolves to multiple polygons.

-guard_band {*x_width y_width*}

Specifies the width of the guard band in the horizontal and vertical direction. The guard band width is the spacing along the boundary of the voltage area where cells cannot be placed. The guard band extends outward from the region. The values specified for this option must be positive integers. When no guard_band is specified, the default values are {0 0}

-target_utilization

Specifies the utilization requested for this voltage area shape during shaping. The value specified must be between 0.1 and 1.0. If this option is not specified, the target_utilization for the design at the time of the voltage area shape creation, whether that value is valid or not, will be used. If the target utilization on the design is changed later on, the value on the voltage area shapes will not be automatically updated.

-cells *cell_list*

Specifies a list of cells to be included in this specific voltage area shape. All cells must be at the same physical hierarchy as the current design. Cells can be physical (leaf) or hierarchical. When specifying hierarchical cells, the cells will be expanded into a set of physical cells that reside under them. If cells are later added or removed from this hierarchical cell, the cell membership of the voltage area shape is not automatically updated. Note that one hierarchical cell can expand into multiple physical cells. In (the unlikely) case that the hierarchical cell is purely logical (when it corresponds to a local module that has no physical representation), the cell will map to an empty set of physical cells, and a warning message will be printed. The specified cells cannot already be explicitly or implicitly associated with another voltage area or voltage area shape.

-exclusive

Sets the voltage area shape to be exclusive with respect to other cells of this voltage area. This option can only be used with the **-cells** option. A voltage area shape should always be treated as an exclusive move bound with respect to cells that do not belong to its voltage area. But to cells associated with its voltage area but not explicitly with this shape, it should be treated as an inclusive move bound. With this option set, the voltage area shape should be treated as exclusive even to cells that belong to its voltage area, so that only cells explicitly associated with this particular shape can be placed within the boundary of the shape, and no other cells, including cells associated with the voltage area, can be placed within this shape.

-merge_regions

Merge into this new shape all shapes within this voltage area that abut or overlap with the input shape. Note that if a shape has explicit cell assignment, that shape will remain discrete and it will not be merged with other shapes in the same voltage area, even if they touches or overlaps. For the same reason, this option cannot be used with the **-cells** option.

-top

Place the voltage area shape to the top of the voltage area shape stack.

-bottom

Place the voltage area shape to the bottom of the voltage area shape stack, right above the first shape of the default voltage area.

-above *voltage_area_shape*

Place the voltage area shape above the given voltage area shape.

-below *voltage_area_shape*

Place the voltage area shape below the given voltage area shape.

DESCRIPTION

The **create_voltage_area_shape** command creates a new shape on an existing voltage area. A voltage area shape can be any rectilinear shape. An optional guard band on the horizontal and vertical direction can also be specified, otherwise the guard band is zero.

The command returns a collection containing the newly created voltage area shape if the command succeeded. Otherwise, the command returns "" if it failed, or TCL_ERROR if there was a command syntax error.

By default, the new voltage area shape is created at the top of the shape position stack (**-top**). With the **-bottom** option, the new voltage area shape will be created at the bottom of the stack, directly above the default voltage area shape. With the **-below** or **-above** option, the voltage area shape will be placed one position below or above an existing voltage area shape.

The **-top**, **-bottom**, **-above**, and **-below** options are mutually exclusive. It is a syntax error to specify more than one of these options at a time. It is an error to place a shape above or below a shape from a different design.

EXAMPLES

The following example constrains the instance INST_1 to lie within the voltage area whose coordinates are lower-left corner (100 100) and upper-right corner (200 200):

```
prompt> create_voltage_area_shape -voltage_area VA1 \
  -region {{100 100} {200 200}} -guard_band {1 2}
{VOLTAGE_AREA_SHAPE_1}
```

The following example creates a voltage area shape above an existing shape

```
prompt> create_voltage_area_shape -voltage_area VA1 \
  -region {{100 100} {200 200}} -above VOLTAGE_AREA_SHAPE_1
{VOLTAGE_AREA_SHAPE_2}
```

This example creates a voltage area shape at the bottom of the position stack, above the default voltage area shape.

```
prompt> create_voltage_area_shape -voltage_area VA1 \
  -region {{300 300} {400 400}} -bottom
{VOLTAGE_AREA_SHAPE_3}
```

SEE ALSO

create_voltage_area(2)
get_voltage_area_shapes(2)
get_voltage_areas(2)
remove_voltage_area_shapes(2)
remove_voltage_areas(2)
report_voltage_areas(2)
set_voltage_area_shape(2)

create_vtcell_fillers

Inserts filler cells based on the threshold voltage types of the cells on the left and right of the gap to be filled.

SYNTAX

status **create_vtcell_fillers**
[-prefix *cell_prefix*]
[-separator *cell_separator*]
[-voltage_area *voltage_areas*]
[-region *region_list*]
[-boundary *region_list*]
[-clear_vt_information]

Data Types

cell_prefix string
cell_separator string
voltage_areas collection
region_list list

ARGUMENTS

-prefix *cell_prefix*

Specifies the prefix for the inserted filler cells.

By default, no prefix is added.

-separator *cell_separator*

Specifies the separator character that is used in the instance name of the inserted filler cells.

The filler cell instance name consists of a prefix, the library reference cell name, and a unique integer. This naming convention allows you to use pattern matching to select the filler cells.

The default separator is "!".

-voltage_area *voltage_areas*

Specifies the voltage areas in which to insert filler cells.

By default, the command inserts filler cells in all voltage areas.

-region *region_list*

Specifies the regions in which to insert filler cells. You can specify one or more rectangles or rectilinear polygons.

Each rectangle is specified by the coordinates of its lower-left and upper-right corners:

{{*llx lly*} {*urx ury*}}

Each polygon is specified by its points:

{{*x1 y1*} {*x2 y2*} ... {*xn yn*}}

-boundary *region_list*

Specifies the regions in which to insert filler cells. You can specify one or more rectangles or rectilinear polygons.

Each rectangle is specified by the coordinates of its lower-left and upper-right corners:

{{*llx lly*} {*urx ury*}}

Each polygon is specified by its points:

{{*x1 y1*} {*x2 y2*} ... {*xn yn*}}

-clear_vt_information

Clears all stored threshold voltage information, including the table with the standard cell threshold voltage types. When you use this option, the command does not perform filler cell insertion.

DESCRIPTION

The **create_vtcell_fillers** command fills the gaps in the block by inserting filler cells based on the threshold voltage type of the cells on the left and right of the gap.

EXAMPLES

The following example fills the block with the correct threshold voltage filler cells.

prompt> **create_vtcell_fillers -prefix "my_fill"**

SEE ALSO

create_stdcell_fillers(2)
set_vt_filler_rule(2)
set_cell_vt_type(2)

create_wire_matching

Creates an analog constraint group with matched_wire intent in the current design.

SYNTAX

collection **create_wire_matching**
[*constraint_group_name*]
-for *objects*
-tolerance *float*
[-match_type length | length_per_layer]
[-relative]
[-exclude *objects*]
[-driver *objects*]
[-force]

Data Types

constraint_group_name string
objects collection
float float

ARGUMENTS

constraint_group_name

Specifies the name of the constraint group.

If you do not specify this argument, the command assigns a system-generated name.

-for *objects*

Specifies the objects that constitute the analog constraint group. The objects can be nets, pins, ports, bundles, or topology_edges, and can be specified as strings (object names) or collections.

This option is required.

-tolerance *float*

Specifies the tolerance within which the constituent objects of the constraint group should match for the specified property.

By default, the command interprets the specified value as an absolute value. To specify the value as a percentage, use the **-relative** option.

This option is required.

-match_type length | length_per_layer

Specifies the characteristic that needs to be matched for all the constituent objects of the constraint group.

The default is **length**.

-relative

Interprets the tolerance value specified by the **-tolerance** option as a percentage.

If you do not specify this option, the tolerance value is interpreted as an absolute number.

-exclude *objects*

Specifies the objects that need to be excluded from the analog constraint group. The objects can be pins or ports, and can be specified as strings (object names) or collections. This option is useful when user gives a net which is connecting physically (across logical hierarchies) to multiple pins / ports in **-for** option but want to exclude a few pins / ports connecting to that net.

-driver *objects*

Specifies the object to be treated as the driver for the analog constraint group. The object can be a pin or port, and can be specified as a string (object name) or collection. This option is useful when user gives a net which is connecting physically (across logical hierarchies) to multiple pins / ports in **-for** option but want to match each driver to receiver connection.

-force

Deletes an existing constraint group with the same name and creates a new constraint group.

If you do not specify this option, the command fails if a constraint group exists with the specified name.

DESCRIPTION

The **create_wire_matching** command creates a constraint group for nets, pins, ports, bundles, and topology_edges with matched_wire intent. Constraint applicable on pin/port check the net which is connecting pin/port to driver pin/port. The constraint is used by the Custom Router tool.

By default, the total length (**-match_type length**) of the objects is compared. You can also match the length per layer (**-match_type length_per_layer**). The selected criteria must match within the tolerance specified by the **-tolerance** option.

EXAMPLES

The following example creates a constraint group named abc for nets and bundles that match the pattern pr* with an absolute tolerance of 10 to match length of each constituent objects. The pr* nets are routed with the Custom Router tool.

```
prompt> create_wire_matching abc -for pr* -tolerance 10
{abc}
```

The following example creates a constraint group named def for nets and bundles that match the pattern pr* with a tolerance of 10 percent to match the length of each constituent objects. The pr* nets are routed with the Custom Router tool.

```
prompt> create_wire_matching def -for pr* -tolerance 10 -relative
{def}
```

SEE ALSO

get_constraint_groups(2)
remove_constraint_groups(2)
report_constraint_groups(2)

cross_probing_filter

Filters a collection based on file name and line number criteria.

SYNTAX

collection **cross_probing_filter**
-source *source_patterns*
objects

Data Types

source_patterns string
objects collection

ARGUMENTS

-source *source_patterns*

This string is the matching criteria used for filtering. Filter a collection by specifying a source file and line number in the "file_pattern[:line_pattern]" format. You can specify multiple patterns by delimiting them with a space character, such as "file_pattern1[:line_pattern1] file_pattern2[:line_pattern2] ...". Objects that match any of the specified file/line patterns are returned in the resulting collection.

The asterisk (*) and question mark (?) wildcard patterns are allowed in the pattern.

objects

Specifies the base collection to be filtered. The base collection may contain cell or port or module or block objects.

DESCRIPTION

Allows you to filter a collection by specifying a source file and line number. This command functions similar to **filter_collection** except the filtering criteria is specific to cross-probing.

You must have a design that was analyzed and elaborated or synthesized using this tool. The tool provides information about where the cell or port or module or block was created, specifying which file and line number the object came from. These objects can be filtered based on file name and/or line number.

Some objects in the design might have been merged, resulting in an object that can potentially have multiple source locations. In this case, the object is returned in the result collection if any of its source positions match any of the specified source patterns specified by *source_patterns*.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the result from the **cross_probing_filter** command, returning cells that originated from a source file ending in m.v, and on line 21:

```
prompt> cross_probing_filter [get_cells -hierarchical] -source "*m.v:21"
{mid1/bot1/c1 mid1/bot2/c1}
```

The following example shows the result from the **cross_probing_filter** command, returning modules that originated from a source file ending in top.v, and on line 32:

```
prompt> cross_probing_filter [get_modules] -source "*top.v:32"
{low}
```

The following example shows the result from the **cross_probing_filter** command, returning blocks that originated from a source file ending in top.v, and on line 1:

```
prompt> cross_probing_filter [get_blocks] -source "*top.v:1"
{top}
```

SEE ALSO

report_cross_probing(2)
get_cross_probing_info(2)

current_block

Sets or gets the current block by block name.

SYNTAX

collection **current_block**
[-quiet]
collection Collection containing the block or "".
block Block name in [libName:]designName[/labelName][.viewName] format or a collection of one block.

string *lib_name*
string *block_name*
string *label_name*
string *view_name*

ARGUMENTS

-quiet

Suppresses all messages.

block

Block object or name.

lib_name

The library name prefix, followed by a colon.

block_name

The name of the block to make the current block.

label_name

The label name, preceded by a slash.

view_name

The view name suffix, preceded by a dot.

DESCRIPTION

Sets or gets the current working block for many commands. Without arguments, **current_block** returns a collection containing the current working block. The combination of the current block and current instance defines the context for many commands.

The block name can be a simple name, like "TOP". Or a block of a specific label, like "TOP/routed". Or a block with a view suffix, like "TOP.design", or "TOP.abstract". Or a block with a library prefix like "lib1:TOP". Or a block with both a library, label, and view name like "workLib:TOP/routed.outline". The library can contain a path to the library, like "/libs/refLib:TOP.design". If the lib name is not given, the current library is used. If the view name is not given, the design view is used.

A block must be opened and read into memory before it can be set to be the current block.

If more than one block matches the given name, an error is issued.

To get a collection of blocks currently available in the tool, use **get_blocks**.

The **current_block** command is different that the **current_design** command in that the **current_design** command looks for a block with a given top module name, whereas the **current_block** command looks for a block with the given block name.

EXAMPLES

The following example uses **current_block** to show the current context and to change the context from one block to another.

```
prompt> current_block
{lib1:TOP.design}

prompt> current_block ADDER.abstract
{lib1:ADDER.abstract}

prompt> current_block
{lib1:ADDER.abstract}

prompt> current_block lib1:TOP/myLabel.design
{lib1:TOP/mylabel.design}
```

The **current_block** command can be used as a parameter to other commands. In the following example, **remove_blocks** is used to delete the working block from memory.

```
prompt> current_block
{lib1:TOP.design}
prompt> remove_blocks [current_block]
Removing block 'lib1:TOP.design'...
1
prompt> current_block
Error: Current block is not defined. (DES-001)
prompt>
```

SEE ALSO

current_instance(2)
current_design(2)
get_blocks(2)
open_block(2)

current_corner

Sets a corner in the current session to be current. Operating condition and parasitics configuration commands apply to the current corner.

SYNTAX

int **current_corner**
[*corner_name*]

Data Types

corner_name string

ARGUMENTS

corner_name

Name of a defined corner in the current design.

DESCRIPTION

This command sets one corner in the current session to be current. Operating condition and parasitics configuration commands apply to the current corner.

Corners are used together with modes, which are used for different operating modes and power schemes. Timing analysis is actually done for various mode/corner combinations, as configured by the **set_scenario_status** command.

Multicorner-Multimode Support

This command works on the current corner.

EXAMPLES

In the following example, two non-default corners are created, then the current_corner is set to "LowPower1".

```
prompt> create_corner LowPower1
1
prompt> create_corner HighTemperature1
```

```
1
prompt> current_corner LowPower1
{"LowPower1"}
prompt> source LowPower1_opconds.tcl
1
```

SEE ALSO

create_corner(2)
create_corner(2)
create_mode(2)
remove_corners(2)
set_scenario_status(2)
source(2)

current_design

Gets the current block or sets the current block to the block containing a specified top-level logic module.

SYNTAX

collection **current_design**
[-quiet]
[*design_name*]

Data Types

design_name string

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

design_name

The name of a design, the top-level logic module of an open block. The command sets the current block to the block containing the specified design. If you do not specify a design name, the command returns a collection containing the current design.

DESCRIPTION

This command sets the current block to the block containing a specified top-level logic module, and returns that block as a collection.

When used without any argument, the command works exactly the same as the **current_block** command; it returns a collection containing the current block. When used with an argument, the argument must be design (a top-level module of an open block). To get a list of designs that you can use as an argument, use the **get_designs** command. The **current_design** command then sets the block containing the specified design as the current block, and returns a collection containing that block.

The **current_block** takes a block as an argument, whereas the **current_design** command takes a top-level module of an open block as an argument. However, both commands return a collection containing the current block. The combination of the current block and the current instance (set by the **current_instance** command) defines the context for many commands.

EXAMPLES

The following examples demonstrate usage of the **current_design** command.

```
prompt> current_block
{libLmp:LEON3mp/fp.design}

prompt> current_design
{libLmp:LEON3mp/fp.design}

prompt> get_blocks
{libLmp:LEON3mp/fp.design}

prompt> get_designs
{LEON3mp mul32 leon3s leon3s_2 leon3s_3}

; Set current block to the block containing top module leon3s_2
prompt> current_design leon3s_2
{libLEON3s:leon3s_2/fp.design}

prompt> current_block
{libLEON3s:leon3s_2/fp.design}

prompt> current_design
{libLEON3s:leon3s_2/fp.design}
```

SEE ALSO

current_block(2)
current_instance(2)
get_designs(2)

current_dft_partition

Sets or gets the design partition for the current specification.

SYNTAX

status **current_dft_partition**
[*design_partition_label*]

Data Types

design_partition_label string

ARGUMENTS

design_partition_label

Specifies the working design partition for many commands. If the *design_partition_label* is not specified, the tool uses the last design partition specified. If the *design_partition_label* specifies a partition name that has not been specified previously using the **define_dft_partition** command, an error is issued and the working design partition remains unchanged.

DESCRIPTION

The **current_dft_partition** command sets the design partition for many specification commands, such as **set_dft_signal**, **set_scan_configuration**, **set_scan_compression_configuration**, and **set_scan_path**. When no arguments are specified, **current_dft_partition** returns the name of the current DFT partition.

To display the names of all of the design partitions in the test model for the current design, use the **report_dft -partitions** command.

EXAMPLES

The following example sets the required number of scan chains to 3 and 2 to the partitions *part1* and *part2*, respectively:

```
prompt> define_dft_partition part1 -include {U1 U2}

prompt> define_dft_partition part2 -include {U3 U4}

prompt> current_dft_partition part1
```

```
prompt> set_scan_configuration -chain_count 3

prompt> current_dft_partition part2

prompt> set_scan_configuration -chain_count 2

prompt> preview_dft
1
```

SEE ALSO

define_dft_partition(2)
preview_dft(2)
report_dft(2)
set_dft_signal(2)
set_scan_compression_configuration(2)
set_scan_configuration(2)
set_scan_path(2)

current_instance

Sets the working instance object and enables other commands to be used relative to a specific instance in the design hierarchy.

SYNTAX

string **current_instance**
[*instance*]

Data Types

instance string

ARGUMENTS

instance

Specifies the working instance (cell). If *instance* is not specified, the focus returns to the top level of the hierarchy in the current design. If *instance* is ".", the current instance remains unchanged. If *instance* is "..", the context is moved up one level in the instance hierarchy. If *instance* begins with "/", the tool sets both the current design and the current instance. More complex examples of *instance* arguments are described in EXAMPLES.

DESCRIPTION

The **current_instance** command sets the working instance. An instance is a cell in the hierarchy of a design. This command differs from **current_design**, which changes the working design, then sets the current instance to the top level of the new current design. The combination of the current design and current instance defines the context for many commands.

current_instance traverses the design hierarchy similar to the way the UNIX **"cd"** command traverses the file hierarchy. **current_instance** operates with a variety of *instance* arguments:

- If no *instance* argument is specified, the focus of the tool is returned to the top level of the hierarchy.
- If *instance* is ".", the current instance is returned and no change is made.
- If *instance* is "..", the current instance is moved up one level in the design hierarchy.
- If *instance* is a valid cell (or cell name) at the current level of hierarchy, the current instance is moved down to that level of the design hierarchy.
- Multiple levels of hierarchy can be traversed in a single call to **current_instance** by separating multiple cell names with slashes. For example, **current_instance U1/U2** sets the current instance down two levels of hierarchy.

- The ".." directive can also be nested in complex *instance* arguments. For example the command **current_instance "../../MY_INST"** attempts to move the context up two levels of hierarchy, then down one level to the **"MY_INST"** cell.

EXAMPLES

The following example uses **current_instance** to move up and down the design hierarchy.

```
prompt> current_design TOP
{"TOP"}

prompt> current_instance U1
U1

prompt> current_instance "."
U1

prompt> query_objects [all_instances ADDER]
{"U1", "U2", "U3", "U4"}

prompt> current_instance U3
U1/U3

prompt> current_instance "../U4"
U1/U4

prompt> current_instance
Current instance is the top-level of block 'Alib:TOP.design'.
```

In the following example, changing the **current_design** resets the **current_instance** to the top level of the new design hierarchy.

```
prompt> current_design
{"TOP"}

prompt> current_instance "U2/U1"
U2/U1

prompt> current_design ADDER
{"ADDER"}

prompt> current_instance .
Current instance is the top-level of block 'Alib:ADDER.design'.
```

The following example uses **current_instance** to go to an instance of another design. The current_design is set by the new design whose name is the name after the first slash of the given instance name.

```
prompt> current_design
{"TOP"}

prompt> current_instance U1
U1

prompt> current_instance "/TOP/U2"
U2
```

```
prompt> current_instance "/ALARM_BLOCK/U6"
U6

prompt> current_design
{"ALARM_BLOCK"}
```

SEE ALSO

current_design(2)
list_blocks(2)
query_objects(2)

current_lib

Sets or returns the current library.

SYNTAX

collection **current_lib**
[-quiet]
[*library_name*]

Data Types

library_name string

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

library

Specifies the working or focal library for many commands. If the *library_name* is not specified, the **current_lib** command returns a collection containing the current library. If *library_name* refers to a library that cannot be found, an error is issued and the working library remains unchanged.

DESCRIPTION

The **current_lib** command gets or sets the current library.

Many library-related commands use the current library by default. The current library is automatically set when the **create_lib** command creates a new library or the **open_lib** command opens a library.

This command returns the current library if successful, or 0 if not. If an illegal name is given, a Tcl error is issued.

EXAMPLES

The following example returns the current library in the current session:

```
prompt> current_lib
{"design"}
```

The following example sets the current library in the current session:

```
prompt> current_lib top_design
{"top_design"}
```

SEE ALSO

open_lib(2)
create_lib(2)
save_lib(2)
close_lib(2)
copy_lib(2)
move_lib(2)
get_libs(2)
set_ref_libs(2)
search_path(3)

current_mode

Sets a mode in the current session to be current. Constraint creation and modification commands apply to the current mode.

SYNTAX

integer **current_mode**
[*mode_name*]

Data Types

mode_name string

ARGUMENTS

mode_name

Name of a defined mode in the current design.

DESCRIPTION

This command sets one mode in the current session to be current. Constraint creation and modification commands apply to the current mode.

Modes are used together with corners, which contain operating condition and parasitics parameters. Timing analysis is actually done for various mode/corner combinations, as configured by the **set_scenario_status** command.

Multicorner-Multimode Support

This command works on the current mode.

EXAMPLES

In the following example, two non-default modes are created, then the current_mode is set to "Mission1".

```
1
prompt> create_mode Mission1
1
prompt> create_mode Test1
```

```
1
prompt> current_mode Mission1
{"Mission1"}
prompt> source Mission1_constraints.tcl
1
```

SEE ALSO

create_corner(2)
create_mode(2)
remove_modes(2)
set_scenario_status(2)
source(2)

current_scenario

Sets the specified scenario as the current scenario or returns a collection that contains the current scenario.

SYNTAX

collection **current_scenario**
[*scenario_name*]

Data Types

scenario_name string

ARGUMENTS

scenario_name

Specifies the scenario to set as the current scenario. The named scenario must already be defined in the current design.

If you do not specify this argument, the command returns a collection that contains the current scenario.

DESCRIPTION

The behavior of the **current_scenario** command depends on whether you specify an argument.

- If you specify a scenario, the command sets this scenario as the current scenario.

  The scenario's mode becomes the current mode, and its corner becomes the current corner. By default, constraint creation and modification commands apply to the current scenario, mode, or corner.

- If you do not specify a scenario, the command returns a collection that contains the current scenario.

If you change the current mode or corner with the **current_mode** or **current_corner** commands, the tool sets the current scenario to whatever scenario has that mode and corner. If there is no such scenario, the current scenario becomes unset.

Multicorner-Multimode Support

This command works on the current scenario.

EXAMPLES

The following command returns the current scenario.

```
prompt> current_scenario
{m1@c1}
```

The following command sets the current scenario to m2@c2.

```
prompt> current_scenario m2@c2
{m2@c2}
```

SEE ALSO

create_corner(2)
create_mode(2)
create_scenario(2)
current_corner(2)
current_mode(2)
remove_corners(2)
remove_modes(2)
remove_scenarios(2)
report_scenarios(2)

current_test_mode

Sets or gets the current working test mode for the current design.

SYNTAX

status **current_test_mode**
[*test_mode_label*]

Data Types

test_mode_label string

ARGUMENTS

test_mode_label

Specifies the working test mode for many commands.

If the *test_mode_label* is not specified, the tool prints the current test mode.

DESCRIPTION

The **current_test_mode** command sets the current working mode. This is the default test mode used for many DFT commands, such as **set_dft_signal** and **dft_drc**.

To display the names of all the modes in the test model data for the current design, use the **list_test_models** command.

To display the names of all the test modes of the current design defined with the **define_test_mode** command, use the **list_test_modes** command.

EXAMPLES

The following example sets the test mode to "Internal_scan" for the current design:

```
prompt> current_test_mode Internal_scan
1
```

The following example gets the test mode for the current design:

```
prompt> current_test_mode
Current test mode is 'Internal_scan'.
1
```

SEE ALSO

current_design(2)
dft_drc(2)
list_test_modes(2)
list_test_models(2)

current_topology_plan

Reports or sets the current topology plan.

SYNTAX

int **current_topology_plan**
[*topology_plan*]

Data Types

topology_plan list

ARGUMENTS

topology_plan

Specifies the topology_plan to set as the current topology_plan.

RETURN VALUE

The current topology_plan.

DESCRIPTION

This command reports the topology_plan that is set as the "current" topology plan in the current block. When a topology_plan is the "current" plan, then topology node and edge creation commands will create the new topology objects in that plan unless otherwise specified. Also, the querying of topology objects will operate under the context of the current topology plan in terms of name lookups.

If the *topology_plan* argument is specified, then the current topology_plan will be changed to the specified plan. Also, the current topology_plan is automatically changed to newly created topology_plans. If the current topology_plan is removed, then there will be no current topology_plan until another plan is set or created.

Changing the current block will clear the current topology plan, as the block context has changed.

EXAMPLES

The following example gets the current topology_plan.

```
prompt> current_topology_plan
{plan0}
```

SEE ALSO

create_topology_plan(2)
get_topology_plans(2)
remove_topology_plans(2)
report_topology_plans(2)

cut_rows

Splits site rows from the current design.

SYNTAX

status **cut_rows**
  -all
  -within *coordinates*

Data Types

coordinates list

ARGUMENTS

-all

Removes all site rows in the current design. This option and the *-within* option are mutually exclusive.

-within *coordinates*

Removes all rows in the specified region. For a rectangular region, specify the coordinates for the lower-left and upper-right corners as {{ll_x ll_y} {ur_x ur_y}}. For a rectilinear region, specify a list of rectilinear coordinates in either clockwise or counterclockwise order. This option and the *-all* option are mutually exclusive.

DESCRIPTION

This command removes rows from the current design. If you specify the **-all** option, the command removes all rows from the design which are not a part of site arrays.

If you specify the **-within** option, the command removes rows within the specified region which are not a part of site arrays. So if a row partially comes under the specified region, then portion of the row falls under that specified region will be removed and remaining portion(s) will form new site rows depending on the number of parts a row gets cut into.

The unbound site rows (site rows whose site def is removed) can be deleted but not modified. Modification of a site row requires parameters of its site definition which is missing in case of an unbound site row.

EXAMPLES

The following example uses the **-all** option to remove all rows from the current design.

```
prompt> cut_rows -all
1
```

The following example uses the **-within** option to cut rows in the specified rectangular region:

```
prompt> cut_rows -within {{3 0} {6 9}}
1
```

The following example uses the **-within** options to cut rows in the specified rectilinear region:

```
prompt> cut_rows -within {{3 12} {3 16} {9 16} {9 20} {12 20} {12 15} {6 15} {6 12}}
```

SEE ALSO

remove_site_rows(2)

date

Returns a string containing the current date and time.

SYNTAX

string **date**

DESCRIPTION

The **date** command generates a string containing the current date and time, and returns that string as the result of the command. The format is fixed as follows:

ddd mmm nn hh:mm:ss yyyy

Where:

ddd is an abbreviation for the day
mmm is an abbreviation for the month
nn is the day number
hh is the hour number (24 hour system)
mm is the minute number
ss is the second number
yyyy is the year

The **date** command is useful because it is native. It does not fork a process. On some operating systems, when the process becomes large, no further processes can be forked from it. With it, there is no need to call the operating system with **exec** to ask for the date and time.

EXAMPLES

The following command prints the date.

```
prompt> echo "Date and time: [date]"
Date and time: Thu Dec  9 17:29:51 1999
```

SEE ALSO

exec(2)

debug_script

Debug a script using the TclDevKit Tcl debugger.

SYNTAX

string **debug_script** script [*port*] [*hostname*]

ARGUMENTS

script

This specifies the Tcl script(s) to be debugged. This script will be sourced and instrumented for debugging.

port

Specifies the port number that the debugger is listening on for a remote debugging application. By default the debugger will be launched on an available port.

hostname

Specifies the host where Prodebug is already running. This is used to connect to an existing remote Prodebug session.

::env(SNPS_TCLPRO_HOME)

This Tcl variable is used to specify the root of the TclPro installation. It is initialized from the environment variable SNPS_TCLPRO_HOME in the user's environment.

USING THE PACKAGE

The **debug_script** command is in the snpsTclPro Tcl package, and all of the commands in this package are defined in the namespace snpsTclPro. The way to use this command is to load the package, which will import the commands it exports into the global namespace. This is shown below.

```
shell> package require snpsTclPro
1.0
```

These commands can be added to the .synopsys setup file for the system, user, or current directory, or the commands can be typed when the pacakge is needed.

DESCRIPTION

The **debug_script** command simplifies the use of the TclDevKit Tcl debugger available from ActiveState (http://www.activestate.com). First it ensures a connection to the Prodebug debugger either by connecting to a running debug session (given a *hostname* and *port*), or by launching the debugger on *localhost* using an available port, and then connecting to it. If there is already an active connection to the debugger then that debugger will simply be used.

Once the connection to the debugger is set up, the specified script will be sourced and debugged. The default behavior for the debugger is to stop at the first line of the source'd script.

The debugger must be run using a "remote" debugging project if you want to connect the Synopsys tools to a running prodebug session. The port number in use by the debugger can be viewed via the Application tab of the "File->Project" Settings dialog.

EXAMPLES

```
# Launch the debugger and then debug the script myscript.tcl.
shell> debug_script myscript.tcl
Selected port 2576 on host localhost
launching prodebug

# now debug another script using the same session
shell> debug_script anotherScript.tcl

# debug myscript.tcl, connecting to the specified debugger session or
# creating one if the connection fails.
shell> debug_script myscript.tcl 2576 localhost
```

CAVEATS

The debugger currently kills the process that started the debugger when you exit, despite the preferences being set differently in the debugger.

When the Synopsys tool has spawned a debugger, that debugger must be exited before the Synopsys tool can exit. If this is not the case then the Synopsys tool will hang on exit, until the debugger is exited.

The **debug_script** command waits for the debug process to start up, before it tries to connect to the debugger. If **debug_script** times out before the debugger is started due to machine or network loading, simply close the debugger that was launched, and then increase the timeout setting with the command **set_debugger_start_timeout**, and then re-run the debugger. The **set_debugger_start_timeout** command takes a single argument which is the value of the timeout in miliseconds, and the default value for the timeout is 10000.

SEE ALSO

check_script(2)
package(2)
namespace(2)

decompose_shape_patterns

Decompose non optimal shape patterns to optimal shape patterns in the current design.

SYNTAX

collection **decompose_shape_patterns**
[-verbose]
[-force]
shape_pattern_list

Data Types

shape_pattern_list collection

ARGUMENTS

-verbose

Display additional debugging information.

-force

Ignores the locked status of objects. If this option is not provided and any of the affected objects has locked status, command will generate a Tcl error and exit.

shape_pattern_list

Specifies a list of non optimal shape patterns to convert and replace with another set of optimal and spatially nearby shape patterns. The list can contain via collections specified with the **get_shape_patterns** command.

DESCRIPTION

This command creates a new shape patterns to replace the specified non optimal shape patterns and returns a collection containing the new shape patterns.

EXAMPLES

The following example converts all shape patterns to more optimal shape patterns.

```
prompt> decompose_shape_patterns [get_shape_patterns]
{RECT_PATTERN_0 RECT_PATTERN_1}
```

SEE ALSO

get_shape_patterns(2)

decompose_via_matrixes

Decompose non optimal via matrixes to optimal via matrixes in the current design.

SYNTAX

collection **decompose_via_matrixes**
[-verbose]
via_matrix_list

Data Types

via_matrix_list collection

ARGUMENTS

-verbose

Display additional debugging information.

via_matrix_list

Specifies a list of non optimal via matrixes to convert and replace with another set of optimal and spatially nearby via matrixes with no holes. The list can contain via collections specified with the **get_via_matrixes** command.

DESCRIPTION

This command creates a new via matrix to replace the specified non optimal via matrixes and returns a collection containing the new via matrix.

EXAMPLES

The following example converts all via_matrixes to more optimal via matrixes.

```
prompt> decompose_via_matrixes [get_via_matrixes]
{VIA_MATRIX_0 VIA_MATRIX_1}
```

SEE ALSO

create_via_matrix(2)
get_via_matrixes(2)
remove_via_matrixes(2)

define_antenna_accumulation_mode

Defines an antenna accumulation mode route rule.

SYNTAX

status **define_antenna_accumulation_mode**
[*library*]
[-cut_to_metal]
[-metal_to_cut]
[-cut_to_metal_for_accumulation_area]
[-metal_to_cut_for_accumulation_area]

Data Types

library list

ARGUMENTS

library

Specifies the library to be updated. The value of *library* can be a library name or a one-element collection of a library. The *library* option is not required. The default is to use the current library.

-cut_to_metal

When you specify the **-cut_to_metal** option, cut ratios are accumulated to metal ratios. Support such behavior for antenna mode 2 and mode 5. Default is off.

-metal_to_cut

When you specify the -cut_to_metal option, metal ratios are accumulated to cut ratios. Support such behavior for antenna mode 2 and mode 5. Default is off.

-cut_to_metal_for_accumulation_area

When you specify the **-cut_to_metal_for_accumulation_area** option, cut ratios are accumulated to metal ratios for accumulation area. Support such behavior for antenna mode 3 and mode 6, Default is off.

-metal_to_cut_for_accumulation_area

When you specify the **-metal_to_cut_for_accumulation_area** option, metal ratios are accumulated to cut ratios for accumulation area. Support such behavior for antenna mode 3 and mode 6, Default is off.

DESCRIPTION

This command defines an antenna accumulation mode route rule and stores it in the library. The command returns a status indicating success or failure.

Note: If the options are not specified with the command, it means the options value are FALSE, and the setting(FALSE) will be stored in the library.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command defines antenna accumulation mode route rules.

To accumulate metal and cut ratio for antenna mode 2 and mode 5
```
prompt> define_antenna_accumulation_mode \
        -cut_to_metal -metal_to_cut
1
```

To accumulate metal and cut ratio for antenna mode 3 and mode 6
```
prompt> define_antenna_accumulation_mode \
        -cut_to_metal_for_accumulation_area \
        -metal_to_cut_for_accumulation_area
1
```

To accumulate metal and cut ratio for antenna mode 2, mode 5,
mode 3 and mode 6
```
prompt> define_antenna_accumulation_mode \
        -cut_to_metal -metal_to_cut \
        -cut_to_metal_for_accumulation_area \
        -metal_to_cut_for_accumulation_area
1
```

SEE ALSO

define_antenna_area_rule(2)

define_antenna_area_rule

Defines an antenna area rule for the specified mode and stores it in the library. The rule considers metal area and max antenna ratio to identify large metal areas where electric charge might pool and damage the gate.

SYNTAX

status **define_antenna_area_rule**
-mode *mode*
-max_area *area*
[-diode_distance *distance*]
[*library*]

Data Types

mode string
area float
distance float
library collection

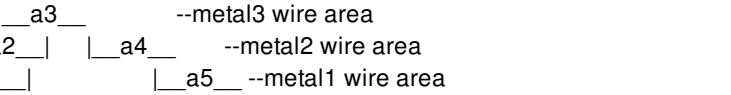
ARGUMENTS

-mode *mode*

Defines the way metal(routing) areas are computed. Valid values are:

- **ignore_lower_layers**: Calculate metal area, ignoring all lower-layer segments.
- **include_lower_layers**: Calculate metal area, including lower-layer segments to the input pins.
- **include_all_lower_layers**: Calculate metal area, including all lower-layer segments.

For example, in the following figure, a1, a2, a3, a4, a5 represent wire areas that connect to the gate. Via connection wire area is omitted in this case.



For different modes, when calculating total routing area for metal3 layer:

- **ignore_lower_layers** : total_routing_area = a3;
- **include_lower_layers** : total_routing_area = a1+a2+a3;
- **include_all_lower_layers** : total_routing_area = a1+a2+a3+a4+a5;

Note that only one rule can be defined for every mode. If two commands contain the same mode number, the second command will overwrite the first one.

-max_area ***area***

This is maximum allowable metal area for a gate, by square distance unit of current library. The metal area (connected to the gate) calculated base on mode can't exceed this constraint.

If this value is zero, the antenna area rule will be ignored.

Valid value: any non-negative number.

-diode_distance ***distance***

This is maximum allowable point to point distance from inserted diode to gate, by distance unit of current library. It is used to prevent diode cell from being inserted far from the gate, so as to guarantee the efficiency of antenna fixing by diode insertion.

If this value is zero, the antenna fixing uses wire rerouting instead of diode insertion.

Valid values are any non-negative number.

library

Specifies the library to define antenna area rule. This can either be a name or a collection of a single library. If not given, the **current_lib** is used.

DESCRIPTION

This command defines an antenna area rule for the specified mode and stores it in the library. The command returns a status indicating success or failure.

EXAMPLES

```
prompt> define_antenna_area_rule -mode ignore_lower_layers \
  -max_area 500 -diode_distance 50.5
```

SEE ALSO

define_antenna_rule(2)
define_antenna_layer_ratio_scale(2)
define_antenna_layer_rule(2)
report_antenna_rules(2)
remove_antenna_rules(2)

define_antenna_layer_ratio_scale

Creates an antenna layer ratio route rule.

SYNTAX

status **define_antenna_layer_ratio_scale**
[*library*]
-layer *layer_name*
-layer_scale *layer_scale*
-accumulate_scale *accumulate_scale*

Data Types

library	list
layer_name	string
layer_scale	float
accumulate_scale	float

ARGUMENTS

library

Specifies the Milkyway library to be updated. The value of *library* can be a library name or a one-element collection of a library. The *library* option is optional. The default is to use the current Milkyway library.

-layer *layer_name*

Specifies the name of the layer.

-layer_scale *layer_scale*

Specifies the layer scale factor.

-accumulate_scale *accumulate_scale*

Specifies the accumulation scale factor.

DESCRIPTION

This command creates an antenna layer ratio route rule. The command returns a status indicating success or failure.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines an antenna layer ratio on the metal1 layer

```
prompt> define_antenna_layer_ratio_scale -layer metal1 \
        -layer_scale 1000.00 -accumulate_scale 2.00
```

SEE ALSO

define_antenna_layer_rule(2)
define_antenna_rule(2)
remove_antenna_rules(2)
report_antenna_rules(2)

define_antenna_layer_rule

Defines an advanced antenna rule for the specified layer.

SYNTAX

status **define_antenna_layer_rule**
[*library*]
-mode *mode*
-layer *layer*
[-name *rule_name*]
[-ratio *ratio*]
[-pratio *pratio*]
[-nratio *nratio*]
-diode_ratio *diode_ratio*
[-scale_factor *scale_factor*]

Data Types

library	collection
mode	int
layer	collection
rule_name	string
ratio	float
pratio	float
nratio	float
diode_ratio	list
scale_factor	list

ARGUMENTS

library

Specifies the library for which to define the antenna layer rule. You can specify either the library name or a collection of a single library.

By default, the command uses the current library.

-mode *mode*

Defines the way antenna areas are computed. The valid values and their descriptions are as follows:

1: Uses the polygon area, ignoring all lower-layer segments.

2: Uses the polygon area, including all lower-layer segments to the input pins.

3: Uses the polygon area, including all lower-layer segments.

4: Uses the side-wall area, ignoring all lower-layer segments.

5: Uses the side-wall area, including all lower-layer segments to the input pins.

6: Uses the side-wall area, including all lower-layer segments.

You can define only one rule for each mode. If two commands contain the same mode number, the second command overwrites the first one.

-layer ***layer***

Specifies the metal or cut layer for which to define the antenna layer rule. You must specify a single layer using either the layer name from the technology file or a collection of a single layer.

-name ***rule_name***

Specifies the name of the created rule.

If you do not specify this option, the command creates or modifies the global antenna layer rule.

-ratio ***ratio***

Specifies the maximum allowable ratio of the antenna area to the gate area if the antenna is not protected by any diode. You specify this value as a floating point number.

-pratio ***pratio***

Specifies the maximum allowable ratio of the antenna area to the p-gate area. You specify this value as a floating point number.

-nratio ***nratio***

Specifies the maximum allowable ratio of the antenna area to the n-gate area. You specify this value as a floating point number.

-diode_ratio ***diode_ratio***

Specifies the allowable ratio of the antenna area to the gate area if the antenna is protected by a diode. You specify this value as a diode ratio vector, which has the following format:

{v0 v1 v2 v3 [v4]}

Each value in the vector is a floating point number. The actual usage of the diode ratio vector depends on the diode mode specified with the **-mode** option. For details, see "Defining Metal Layer Antenna Rules" in the *Fusion Compiler Implementation User Guide*.

-scale_factor ***scale_factor***

Specifies the scale factor of the metal area to the gate area. This option is valid only for diode modes 11 and 12.

DESCRIPTION

This command defines an advanced antenna rule for the specified layer.

The persistency of the rule depends on several factors:

- If you run the command in the library manager, the rule is saved in the cell library and is persistent.
- If you run the command in an implementation tool, the rule is saved in the design library only if all technology data is stored in

the design library and not in a reference library.

However, in most cases, the design library references the technology data in a technology library or a cell library that contains technology data. In these cases, the rule is not saved in the design library and is not persistent.

This command cannot be used with an antenna rule that is defined with diode mode 20 because the diode mode 20 antenna rule is layer independent.

EXAMPLES

The following example defines an advanced antenna rule for the metal1 layer that uses side-wall area and ignores all lower-layer segments.

```
prompt> define_antenna_layer_rule -layer metal1 -mode 4 \
  -ratio 400 -diode_ratio {0.336 -0.5 400 2400}
```

SEE ALSO

define_antenna_rule(2)
get_antenna_rule_names(2)
report_antenna_rules(2)
remove_antenna_rules(2)

define_antenna_rule

Defines an advanced antenna rule for the specified mode and stores it in the library.

SYNTAX

status **define_antenna_rule**
-mode *mode*
-diode_mode *diode_mode*
[-name *rule_name*]
[-metal_ratio *metal_ratio*]
[-cut_ratio *cut_ratio*]
[-metal_pratio *metal_pratio*]
[-metal_nratio *metal_nratio*]
[-cut_pratio *cut_pratio*]
[-cut_nratio *cut_nratio*]
[-protected_metal_scale *metal_scale*]
[-protected_cut_scale *cut_scale*]
[-area_threshold *area*]
[-layer_area_threshold *{min_layer max_layer area}*]
[*library*]

Data Types

mode integer
diode_mode integer
rule_name string
metal_ratio float
cut_ratio float
metal_pratio float
metal_nratio float
cut_pratio float
cut_nratio float
metal_scale float
cut_scale float
area float
layer_area_threshold list
library collection

ARGUMENTS

-mode *mode*

Defines the way antenna areas are computed. The valid values and their descriptions are as follows:

- **1** - Uses a polygon area, ignoring all lower-layer segments.
- **2** - Uses a polygon area, including all lower-layer segments to the input pins.
- **3** - Uses a polygon area, including all lower-layer segments.
- **4** - Uses a side-wall area, ignoring all lower-layer segments.
- **5** - Uses a side-wall area, including all lower-layer segments to the input pins.
- **6** - Uses a side-wall area, including all lower-layer segments.

Note that only one rule can be defined for every mode for a given rule name. If two commands contain the same mode number and same rule name, the second command overwrites the first one.

-diode_mode ***diode_mode***

Defines the protection capability of the diode. By default, all output pins are considered to be a diode. The valid values and their descriptions are as follows:

- **0** - The diodes do not provide any protection.
- **1** - The diodes provide unlimited protection.

  In this mode, the highest metal layer is not checked for antenna violations because the input-to-output connection is completed when the highest metal layer is formed.
- **2** - Diode protection is limited; if more than one diode is connected, the largest value of the maximum antenna ratio for all diodes is used.
- **3** - Diode protection is limited; if more than one diode is connected, the sum of the maximum antenna ratios for all diodes is used.
- **4** - Diode protection is limited; if more than one diode is connected, the sum of the diode-protection values for all diodes is used to compute the maximum antenna ratio.
- **5** - Diode protection is limited; the maximum diode-protection value for all diodes is used to calculate the equivalent gate area.
- **6** - Diode protection is limited; the sum of the diode-protection values for all diodes is used to calculate the equivalent gate area.
- **7** - Diode protection is limited; the maximum diode-protection value for all diodes is used to calculate the equivalent metal area.
- **8** - Diode protection is limited; the sum of the diode-protection values for all diodes is used to calculate the equivalent metal area.
- **9** - Diode protection is limited; scaling is based on maximum diode protection.
- **10** - Diode protection is limited; scaling is based on total diode protection.
- **11** - Diode protection is limited; scaling is based on maximum diode protection.
- **12** - Diode protection is limited; scaling is based on total diode protection.
- **19** - Has well type dependent data in addition to well independent data.
- **20** - Has accumulated area threshold for floating metal or vias.
- **21** - Diode protection is limited; this is a nwell antenna rule which will safe guard by maintaining metal to nwell area ratio.
- **22** - Diode protection is limited; this is a TSV(Through Silicon Via) antenna rule which will safe guard by the ratio of TSV

area to the gate area that are electrically connected.

-name *rule_name*

Specifies the name of the rule to be created. This is an optional option. When not specified, the global-antenna rule is created/modified.

-metal_ratio *metal_ratio*

Specifies the maximum allowable ratio for metal area to gate size if the metal layer is not defined by using the **define_antenna_layer_rule** command. It's required when metal_pratio and metal_nratio are not specified. If this value is zero, the ratio is ignored. The valid value is any non-negative number. This can't be specified for **-diode_mode 20**.

-cut_ratio *cut_ratio*

Specifies the maximum allowable ratio for cut area to gate size if the cut layer is not defined by using the **define_antenna_layer_rule** command. It's required when cut_pratio and cut_nratio are not specified. If this value is zero, the ratio is ignored. The valid value is any non-negative number. This can't be specified for **-diode_mode 20**.

-metal_pratio *metal_pratio*

Specifies the maximum allowable ratio for antenna area (metal) to p-gate area. Use together with metal_nratio, required when metal_ratio is not specified. This can't be specified for **-diode_mode 20**.

-metal_nratio *metal_nratio*

Specifies the maximum allowable ratio for antenna area (metal) to n-gate area. Use together with metal_pratio, required when metal_ratio is not specified. This can't be specified for **-diode_mode 20**.

-cut_pratio *cut_pratio*

Specifies the maximum allowable ratio for antenna area (cut) to p-gate area. Use together with cut_nratio, required when cut_ratio is not specified. This can't be specified for **-diode_mode 20**.

-cut_nratio *cut_nratio*

Specifies the maximum allowable ratio for antenna area (cut) to n-gate area. Use together with cut_pratio, required when cut_ratio is not specified. This can't be specified for **-diode_mode 20**.

-protected_metal_scale *metal_scale*

Specifies the value used to scale the area of the metal layer that is protected by the diode. The option is used only when the mode is 2 or 5. If this value is zero, the scale is ignored. The valid value is any non-negative number. By default, the value is set to 1.0.

-protected_cut_scale *cut_scale*

Specifies the value used to scale the area of the cut layer that is protected by a diode. The option is used when mode is 2 or 5 only. If this value is zero, the scale is ignored. The valid value is any non-negative number. By default, the value is set to 1.0.

-area_threshold *area*

Specifies the threshold for accumulated area of floating metal or via. This can be specified only with **-diode_mode 20**.

-layer_area_threshold *{min_layer max_layer area}*

Specifies the threshold for the layers specified by the range {min_layer max_layer} for accumulated area of floating metal or via. This can be specified only with **-area_threshold**. Max_layer's layer_number should not be lesser than min_layer's layer_number.

DESCRIPTION

This command defines an advanced antenna rule for the specified mode and stores it in the library.

With the use of the **-name** option, the antenna rule definitions should be unique with respect to the values specified with the parameters. If an antenna rule with configurations similar to the one being created, already exists, then the command reports the same and stops processing. Multiple invocations of the command with different configurations and same rule-name will overwrite the previous configuration associated with the given rule-name.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the use of the command specifying a polygon area (ignoring all lower-layer segments), limited diode protection, maximum allowable ratio for metal area to gate size of 1000, and a maximum allowable ratio for cut area to gate size of 0.

```
prompt> define_antenna_rule -mode 1 -diode_mode 2 \
  -metal_ratio 1000 -cut_ratio 0
```

The following example shows the use of the command specifying a polygon area (ignoring all lower-layer segments), limited diode protection, maximum allowable ratio for metal area to p-gate size of 1000, maximum allowable ratio for metal area to n-gate size of 500, maximum allowable ratio for cut area to p-gate size of 0 and a maximum allowable ratio for cut area to n-gate size of 0.8. The example also shows the usage of '-name' option:

```
prompt> define_antenna_rule -name m1_dm2_rule1 -mode 1 -diode_mode 2 \
  -metal_pratio 1000 -cut_pratio 0 -metal_nratio 500 -cut_nratio 0.8
```

SEE ALSO

define_antenna_layer_rule(2)
get_antenna_rule_names(2)
report_antenna_rules(2)
remove_antenna_rules(2)

define_dft_design

Characterizes a design as a DFT design to be used for DFT insertion.

SYNTAX

status **define_dft_design**
-design_name *design_name*
[-type *design_type*]
[-interface *access_list*]
[-params *param_list*]

Data Types

design_name string
design_type string
access_list list
model_path_name string
param_list list

ARGUMENTS

-design_name *design_name*

Identifies the design to be instantiated during DFT insertion.

This option is required.

-type *design_type*

Specifies the type of the DFT design to which the design corresponds. Valid design types are:

DECODE CONTROL_FORCE Z_CONTROL_FORCE OBSERV_DGEN
SHIFT_REG BC_1 BC_2 BC_3
BC_4 BC_5 BC_7 BC_8
BC_9 BC_10 WC_D1 WC_D1_S
WC_S1 WC_S1_S INSTRREG TAP
TAP_UC LBIST_CONTROLLER LBIST_CODEC LBIST_XCONTROLLER
LBIST_XCODEC CLK_MUX CLK_CHAIN MBIST_CONTROLLER
MBIST_WRAPPER PAD AC_1 AC_2
AC_7 AC_SELU AC_SELX

There is no default for this option. When the option is not specified, the tool creates a new DFT design type from the specified design name and the specified interface access list.

-interface *access_list*

Specifies the list of signal mapping triplets relating a signal type to a pin of the design specified with **-design_name** *design_name*. The valid format of a signal mapping triplet is as follows:

signal_type_name pin_name pin_polarity

The *signal_type_name* specifies the signal type name of the latest of DesignWare version of the DFT design pin name. The *pin_name* specifies the pin name of the design specified with **-design_name** *design_name*. The *pin_polarity* specifies the polarity of the specified design pin with respect to the *signal_type_name*.

Valid values for *pin_polarity* are **h** for normal polarity and **l** for negative polarity.

There is no default for this option. When the option is not specified, it is assumed that the design does not have interface access pins for use with DFT insertion.

-params *param_list*

Specifies the list of parameter triplets for the specified DFT design. The valid format of a parameter triplet is as follows:

param_name param_type param_value

The *param_name* specifies the name of the parameter, *param_type* specifies the type of the parameter, and *param_value* specifies the value of the parameter. Valid values for *param_type* are **integer**, **float**, **boolean**, and **string**.

These parameters are stored on the specified DFT design for use with DFT insertion.

There is no default for this option.

DESCRIPTION

The **define_dft_design** command characterizes a design as a DFT design of a certain type for use with DFT insertion. Ensure that the specified design functionality matches the standard design of the specified type. The specified design is used to instantiate an IP instead of a DesignWare IP during DFT insertion.

EXAMPLES

The following example instructs the tool to use the *my_des* design as a WC_D1 for wrapper insertion:

```
prompt> define_dft_design -design_name my_des -type WC_D1 \
      -interface {shift_clk capture_clk h capture_en capture_en \
      h shift_en shift_dr h cti si h cto so h cfi data_in \
      h cfo data_out h}
1
```

SEE ALSO

preview_dft(2)

define_dft_partition

Defines a design DFT partition to be created during DFT synthesis.

SYNTAX

status **define_dft_partition**
design_partition_label
[-include *list_of_design_objects*]
\" [-wrapper disable | enable]

Data Types

list_of_design_objects list

ARGUMENTS

design_partition_label

Specifies a user-defined text string containing only alphanumeric characters (a-z, A-Z, 0-9) and the underscore (_).

-include *list_of_design_objects*

Specifies the list of leaf or hierarchical cells, design references, or core scan segments that belong to the defined partition.

You can mix object types in the list. Wildcards and collections are supported.

\" .IP "-wrapper disable | enable" \" Specifies whether the DFT partition should be core-wrapped. \" \" When this feature is enabled, the **-include** object list must contain \" only a single hierarchical cell. The hierarchical boundary of that cell then \" becomes wrappable when the DFT core-wrapping client is enabled. \" \" The default is **false**. \"

DESCRIPTION

The **define_dft_partition** command allows you to define a specific design partition during DFT synthesis. The partition is referenced to the top level in a top-down flow.

The partition information is then used during the DFT architect and insertion process. Each design partition will be DFT-inserted based on its DFT constraints.

You can include clock-gating cells in DFT partition definitions. Although clock-gating cells are not stitched into scan chains, they will use partition-specific scan-enable signals for their test pins (if defined). Clock-gating cells can be included by leaf cell, enclosing hierarchical cell, or clock. When referencing a Power Compiler clock-gating cell by leaf cell, reference the mapped clock-gating cell inside the hierarchical clock-gating cell wrapper.

EXAMPLES

The following example defines 2 design partitions for use in a DFT insertion. The design contains at the top level 4 hierarchical cells named *U1*, *U2*, *U3* and *U4*, respectively.

```
prompt> define_dft_partition part_1\
        -include {U1 U2}

prompt> define_dft_partition part_2\
        -include {U3 U4}
```

SEE ALSO

current_dft_partition(2)
report_dft(2)

define_hdl_library

Maps an HDL library name to a template library location.

SYNTAX

status **define_hdl_library**
hdl_library_name
[-path *path*]

Data Types

hdl_library_name string
path string

ARGUMENTS

hdl_library_name

Specifies the hdl_library_name to be mapped. When specified as "default", the hdl_library_name is taken from **hdlin.hdl_library.default_name**.

-path *path*

Specifies the location of the template library. If the path option is not used, the path is the file join of **hdlin.hdl_library.default_dirname** and the uppercase versions of hdl_library_name.

DESCRIPTION

This command maps an HDL library to a template library location. The template library is not required to exist yet when the command is called. The command does not create template libraries or cause any changes on disk.

EXAMPLES

The following two commands has the same effect:

prompt> **define_hdl_library** my_lib

and

prompt> **define_hdl_library** my_lib **-path** [file join [get_app_option_value -name hdlin.hdl_library.default_dirname] [string toupper

SEE ALSO

analyze(2)
elaborate(2)
report_hdl_libraries(2)
hdlin.hdl_library.default_dirname(3)
hdlin.hdl_library.default_name(3)

define_libcell_subset

Defines a subset of library cells to restrict the optimization of sequential and instantiated combinational cells.

SYNTAX

status **define_libcell_subset**
[-libcell_list *lib_cells*]
[-family_name *name*]

Data Types

lib_cells list
name string

ARGUMENTS

-libcell_list *lib_cells*

Specifies a list of library cells to be a family. These library cells must abide by the following rules: First, the library cells cannot belong to another family. Second, they must have the same functional identification.

-family_name *name*

Specifies a name for the subset of library cells.

DESCRIPTION

The **define_libcell_subset** command defines a subset of library cells to be protected. After the library cells are grouped into a family, they are not available for use during optimization. However, unmapped sequential cells that are specified by the **set_library_subset** command can be mapped to a library cell in the same family. A mapped sequential/combinational cell can also be optimized to another library cell in the same family as its original library cell. A library cell can only belong to one family. Therefore, if you need to change the family of a library cell, the family should be removed first using the **remove_libcell_subset** command.

To remove a family, use the **remove_libcell_subset -family_name** command.

EXAMPLES

In the following example, **define_libcell_subset** groups logic library cells SDFLOP1 and SDFLOP2 into a family called special_flops. Next, **set_libcell_subset** sets the family for unmapped cells u1 and u2 to the defined special cell family, special_flops, which consists of the library cells SDFLOP1 and SDFLOP2. As a result, u1 and u2 can only be mapped to either SDFLOP1 or SDFLOP2.

```
prompt> define_libcell_subset -libcell_list "SDFLOP1 SDFLOP2" -family_name special_flops
prompt> set_libcell_subset -object_list [get_cells u1 u2] -family_name special_flops
```

SEE ALSO

remove_libcell_subset(2)
report_libcell_subset(2)
set_libcell_subset(2)
define_libcell_subset(2)

define_name_rules

Defines a set of naming rules for ports, cells, and nets in the design.

SYNTAX

status **define_name_rules**
rule_name
[-type *object_type*]
[-special *output_format*]
[-equal_ports_nets]
[-inout_ports_equal_nets]
[-collapse_name_space]
[-map *map_string*]
[-check_bus_indexing]
[-check_internal_net_name]
[-remove_irregular_net_bus]
[-remove_irregular_port_bus]
[-allowed *chars_list*]
[-restricted *chars_list*]
[-first_restricted *chars_list*]
[-last_restricted *chars_list*]
[-remove_chars *chars_list*]
[-replacement_char *rep_char*]
[-reserved_words *list_of_words*]
[-case_insensitive]
[-max_length *length*]
[-prefix *string*]
[-add_dummy_nets]
[-dummy_net_prefix *string*]
[-remove_internal_net_bus]
[-remove_port_bus]
[-flatten_port_bus]
[-flatten_multi_dimension_buses]
[-dont_change_bus_members]
[-dont_change_ports]
[-remove_synonym_nets]
[-reset]
[-target_bus_naming_style *bus_naming_style*]

Data Types

rule_name string
object_type string
output_format string
map_string string
chars_list string
rep_char string

list_of_words list
length integer
string string
bus_naming_style string

ARGUMENTS

rule_name

Specifies the name of the rule set being defined. If this is a new rule set name, the command creates a new set of rules. If this is an existing rule set name, the command adds the specified rules to the existing rule set.

-type *object_type*

Specifies that the rules being defined apply only to the given type of objects. Allowed values for object type are **port**, **cell**, **net** or **all**. By default, the defined rules apply to all object types.

-special *output_format*

Specifies a predefined set of rules for a particular output format. The only output format currently supported is **verilog**.

-equal_ports_nets

Specifies that each net connected to an input port or output port must have the same name as the connected port. If a net is connected to more than one port and none of the ports has the same name as the net, the net's name will be changed to one of the port's name. For a feedthrough net, the net name must be the same as the input port name. By default, net names need not match the names of the connected ports. Note that the rename could fail due to collision of names.

-inout_ports_equal_nets

Specifies that each net connected to an inout (bidirectional) port must have the same name as the connected port. This requirement does not apply to a net connected to multiple ports.

-collapse_name_space

Specifies that ports, cells, and nets share the same name space, which means that each name can be used by only one object. For example, a net and a port cannot share the same name, even if they are connected. By default, ports, cells, and nets each have their own name space, so a net and a port can share the same name, even if they are not connected. Applying a collapsed name space causes the **change_names** command to "uniquify" the port, cell, and net names (in that order).

-map *map_string*

Changes specific names or patterns to new names or patterns. You specify the old patterns and the replacement strings in the *map_string* argument. For details, see "Mapping Rules" in the DESCRIPTION section.

-check_bus_indexing

Sorts the index values of port buses in increasing order and bit-blasts the incomplete buses.

-check_internal_net_name

Prohibits internal nets (those not connected to any port) from having the same name as port.

-remove_irregular_net_bus

Bit-blasts any net bus that contains irregular members. Only if all members are of the same name that is different than the bus name then bus name is renamed to its member's name provided there is no existing net or bus net of the same name.

-remove_irregular_port_bus

Bit-blasts any port bus that contains irregular members.

-remove_synonym_nets

Removes all the synonym nets.

-flatten_port_bus

Ensure that all pins of a bused port have names of the form "*bus_name*[*i*]". This will typically rename buses whose HDL type involved VHDL records, SystemVerilog structs, or multidimensional arrays. Although pin names are changed, the bus's HDL type in memory is not. Regular pin names permit write_verilog to netlist the bus as a one-dimensional bus port.

-flatten_multi_dimension_buses

Flattens multi-dimension ports, net buses, and arrays so writing out is easier. Although pin names are changed, the bus's HDL type in memory is not. If **-flatten_port_bus** and **-flatten_multi_dimension_buses** are turned on, the former overrides the latter. If only **-flatten_multi_dimension_buses** is turned on, then, **-remove_irregular_port_bus** would be enabled as well, to allow write_verilog to successfully write out single dimension buses with non-numeric indices.

-allowed *chars_list*

Allows only the listed characters to be used in object names. The list must contain at least ten characters. You can specify the allowed characters individually or as alphanumeric ranges. Disallowed characters are replaced by an underscore character or other character specified with the **-replacement_char** option, or they are removed if you use the **-remove_chars** option. For details, see "Character Restriction Rules" in the DESCRIPTION section. By default, all printable characters are allowed in names.

-restricted *chars_list*

Prevents the listed characters from being used in object names.

-first_restricted *chars_list*

Prevents the listed characters from being used as the first character of each object name.

-last_restricted *chars_list*

Prevents the listed characters from being used as the last character of each object name.

-remove_chars

Causes characters disallowed by the **-allowed** or **-restricted** option to be removed rather than replaced. By default, disallowed characters are replaced by an underscore character or other character specified with the **-replacement_char** option.

-replacement_char *rep_char*

Specifies the character used to replace characters disallowed by the **-allowed** or **-restricted** option. The default replacement character is the underscore (_).

-reserved_words *list_of_words*

Specifies a list of words that cannot be used as object names, such as words considered reserved in your target system. By default, there are no reserved words.

-case_insensitive

Causes the character case (uppercase or lowercase) to be ignored when names are compared. For example, when you use this option, the names "ABC" and "abc" are considered to be equivalent and therefore conflicting within a name space. By default, the name comparison is case-sensitive; lowercase and uppercase characters are considered to be different characters, so there is no conflict between "ABC" and "abc".

-max_length *length*

Specifies the maximum allowed number of characters in each name. The specified maximum length must be at least 8. By default, names of any length are allowed. Setting the *length* argument to 0 resets the rule to the default behavior (any length allowed).

-prefix *string*

Specifies the prefix used for generating new object names. A prefix is required only for renaming excessively long names when truncation and indexing fail to generate a unique name. By default, the prefix is "P" for ports, "U" for cells, and "N" for nets.

-add_dummy_nets

Adds a new dummy net for each unconnected pin and connects it to the pin.

-dummy_net_prefix *string*

Sets the prefix for generating the names of new dummy nets created by the **-add_dummy_nets** option. The default prefix is SYNOPSYS_UNCONNECTED_.

-remove_internal_net_bus

Bit-blasts all internal net buses that do not have any port connection.

-remove_port_bus

Bit-blasts all port buses. Net buses that connect to ports are also bit-blasted.

-dont_change_bus_members

Specifies not to change the elements that are part of a bus. This option has higher priority than all other options that change bus and bus port names.

-dont_change_ports

Specifies not to change ports.

-reset

Resets all rules to their default behavior, which removes all restrictions on object names.

-target_bus_naming_style *bus_naming_style*

Specifies the rule for bus delimiters to be used in port, pin, and net names to denote bus subscripts while write_gds, write_oasis and write_def or while querying through get_ports/pins/nets. Valid values are "[]", "{}", "()", and "<>".

DESCRIPTION

The tool follows certain naming conventions for ports, cells, and nets. When you export a design to an external format, you might need to have the object names follow different naming conventions. To change the names of objects, use the **define_name_rules** command to define the desired naming rules and the **change_names** command to change the names according to these rules.

The rules defined by the **define_name_rules** command can:

- Force the name of any net connected to a port to match the port name
- Force all port, cell, and net names to be unique across all objects

- Change all occurrences of a given string in object names to a new string
- Allow only a given list of characters to be used as the first character, the last character, or all characters of object names
- Prohibit a specified list of characters in object names
- Prohibit a specified list of words as object names
- Restrict the number of characters in each name to a specified maximum

The **change_rules** command enforces these rules by performing one or more of the following actions:

- Deleting characters
- Changing the case of characters
- Replacing characters with an underscore or other specified characters
- Appending a numeric index to the name
- Assigning an entirely new name consisting of a prefix and a numeric index

The **define_name_rules** command defines a set of rules for naming design objects. You assign a name for a set of rules in the command. Each type of rule is supported by a command option. You can use multiple **define_name_rules** command to specify multiple rules for a given rule set.

After you define all the rules for a rule set, you can use the **report_name_rules** command to report the rules, or the **change_names** command to actually change the names in the design according to the rules. To preview the changes to the design without actually performing those changes, use the **report_names** command.

Port and Net Rules

To force the names of nets connected to ports match the names of the connected ports, use the **-equal_ports_nets** option for input and output ports or the **-inout_ports_equal_nets** option for bidirectional ports. If renaming a net would cause a conflict, or if a net is connected to multiple ports, the name is not changed and a warning message is issued.

Name Space Rules

By default, the name spaces of ports, cells, and nets are separate. Names of ports and cells must be unique within a design. A port, a cell, and a net within a design can all share the same name.

To collapse the separate name spaces into a single name space, use the **-collapse_name_space** option. In that case, ports, cells, and nets within a design must be unique across all objects. Conflicting names are modified by appending digits.

If you use both the **-collapse_name_space** and **-equal_ports_nets** options, the **-equal_ports_nets** rule has priority, so when a net is connected to a port, port and net names still match.

Type-Specific Rules

Name rules can be tailored for ports, cells, and nets object types with the **-type** option. The options to **define_name_rules** that can be specified by object type are as follows:

-max_length
-allowed
-restricted
-first_restricted
-last_restricted
-replacement_char
-remove_chars
-prefix

To specify the maximum length restriction on ports different than the maximum length for cells, make multiple calls to **define_name_rules**.

In the following example, the maximum length of ports is set to 12. The maximum length of cells is set to 8. The maximum length of nets is unrestricted.

prompt> **define_name_rules EXAMPLE -max_length 12 -type port**

prompt> **define_name_rules EXAMPLE -max_length 8 -type cell**

Special Rules

The **-special** option invokes a predefined set of rules for a specific output format. Currently, **verilog** is the only supported output format. Using **-special verilog** is equivalent to specifying the following options:

-collapse_name_space
-equal_ports_nets
-inout_ports_equal_nets
-remove_irregular_port_bus
-remove_irregular_net_bus

You can optionally specify other rules, in addition to the ones implicitly invoked by the **-special verilog** option.

Note: The **change_names** command lets you specify **-rules verilog** as the rule name, which is somewhat different from using the **-special verilog** option in the **define_name_rules** command. For details, see the man page for the **change_names** command.

Mapping Rules

The **-map** option changes specific names or patterns to new names or patterns. You specify the old patterns and the replacement strings in the *map_string* argument using the following format:

{{"*pattern*","*replacement*"} [,{"*pattern*","*replacement*"}] ...}

The elements of the mapping string are grouped into pairs. The first element of a pair is a character pattern to be replaced, and the second element is the replacement string. For example,

prompt> **define_name_rules my_map_rule -map {{"a","A"}}**

This mapping rule replaces each instance of a lowercase "a" with an uppercase "A".

There can be any number of pairs in the mapping string. Each pair is enclosed by curly braces and separated from other pairs by a comma. The **change_names** command performs search-and-replace operations in the order specified in the **map_string**, from left to right.

The pattern string also provides limited regular expression capability. The following 4 special characters are supported to make the mapping strings more specific:

- The **$** (dollar sign) indicates the end of string.
- The **^** (caret) indicates the beginning of string.
- The ***** (asterisk) indicates a wildcard matching 0 or more characters.
- The **?** (question mark) indicates a wildcard matching 1 character.

The following example defines a rule that any name ending with **_reg** is replaced by **in** at the end of the string:

prompt> **define_name_rules my_rule0 -map {{"_reg$", "in"}}**

The following example defines a rule that any name beginning with **_reg** is replaced by **in** at the beginning of the string:

prompt> **define_name_rules my_rule1 -map {{"^_reg", "in"}}**

The following example defines a rule that any substring of a name that begins with **_r** and ends with **g** is replaced by the **in** string:

prompt> **define_name_rules my_rule2 -map {{"_r*g", "in"}}**

The following example defines a rule that any substring of a name that begins with **_r** followed any one character, then ending with **g** is replaced by **in**:

prompt> **define_name_rules my_rule3 -map {{"_r?g", "in"}}**

Once a mapping rule is defined, you can only use **-reset** to reset the mapping rule. You cannot redefine the same matching pattern within the same rule. For example, assume you define the following rule:

prompt> **define_name_rules my_rule4 -map {{"A", "X"}}**

If you then decide to replace all occurrences of **A** with **Z**, you must first execute a reset and then redefine the name rules as follows:

prompt> **define_name_rules my_rule4 -reset**

prompt> **define_name_rules my_rule4 -map {{"A", "Z"}}**

For escaping a wild card pattern single backslash \ can be used.

The following example defines a rule that any substring of a name **_r*g** will be matched and replace with **in**, it considers * as a character instead wildcard pattern.

prompt> **define_name_rules my_rule2 -map {{"_r*g", "in"}}**

Order of Applying Name Rules

The **change_names** command performs changes in the following order:

1. Map rules -- Rules that directly change object names, as specified by the **-map** option.
2. Meta rules -- Rules that extend beyond mapping, as specified by the **-collapse_name_space** option.
3. Design rules -- Rules that apply to object names in terms of whole design naming methodology: **-flatten_port_bus**, **-flatten_multi_dimension_buses**, **-equal_ports_nets**, **-inout_ports_equal_nets**, **-check_internal_net_name**, **-remove_irregular_port_bus**, and **-check_bus_indexing**, in that order.

The **change_names** command changes the port names first, then the cell names, and finally the net names. When there is a conflict between a port name and a cell or net name, the port name has higher priority and is not changed. Instead, the cell or net name is changed.

In some cases, later changes can overwrite earlier ones, possibly resulting in unexpected changes. To control the order in which naming rules are applied, you can create a separate rule set for each rule using a separate **define_name_rules** command, and assigning a different rule set name to each rule. Then you can apply each rule separately using its own **change_names** command in the desired order. Perform the least important changes first and the most important changes last.

Character Restriction Rules

You can restrict the usage of characters in object names by using the **-allowed**, **-restricted**, **-first_restricted**, and **-last_restricted** options.

To specify the characters allowed or disallowed, use the **-allowed** or **-restricted** option. Either of these options overrides any previous settings made with the **-allowed**, **-restricted**, **-first_restricted**, and **-last_restricted** options.

In conjunction with the **-allowed** or **-restricted** options, you can optionally use the **-first_restricted** and **-last_restricted** options to further restrict the characters for the first and last character of each name.

The argument of these options is a string that specifies the characters that can or cannot be used in object names. The hyphen character indicates a range of ASCII characters, for example, the string "a-k m 1-5" means all the letters from "a" to "k", the single letter m, and the numerals 1 through 5. To specify a literal hyphen character, precede it with two backslashes (\\-). Space characters are ignored. At least 10 different characters must be allowed. Multiple characters or ranges can be specified in any order.

The following example specifies a character set consisting of only the uppercase letters, numerals, and underscore character.

prompt> **define_name_rules UC_ONLY -allowed "A-Z 0-9 _"**

In the following example, some punctuation characters are disallowed, and the lowercase letters are disallowed as the first character of a name.

prompt> **define_name_rules NOPUNC** \
-restricted "!@#$%^&*()\-" -first_restricted "a-z"

The hyphen character (-) is disallowed by the character sequence \-. The quotation marks character (") is disallowed by the sequence \".

If an object name contains a disallowed character, the **change_names** command performs the following steps to modify the name:

1. Removes the character if the **-remove_chars** option was used in the rule definition.

2. Changes the case. For example, if lowercase letters are not allowed and uppercase characters are allowed, the name "sum4" is changed to "SUM4".

3. Replaces the character with an underscore character (by default) or the character specified by the **-replacement_char** option. For example, if punctuation characters are not allowed and the replacement character is an underscore, the name "U$1" is changed to "U_1".

4. Appends a number the name if necessary to make the name unique. If the previous steps create duplicate names, a number is appended to the new names to make them different.

Reserved Word Rules

To prevent specific names from being used as object names, specify the disallowed words using the **-reserved_words** option. For example, the following command prevents the words START and END from being used as names:

prompt> **define_name_rules MYWORDS -reserved_words {"START", "END"}**

Case Sensitivity Rule

By default, the naming rules are case-sensitive. The uppercase and lowercase versions of a letter are considered different characters.

To consider uppercase and lowercase letters as equivalent, use the **-case_insensitive** option in the rule definition. In that case, the words "END", "End", and "end" are all considered the same, so if the word "END" is disallowed, then "End and "end" are also disallowed.

Maximum Length Rule

To specify a maximum length for names, use the **-max_length** option to specify the maximum number of characters per name.

If an object name contains more than this number of characters, the **change_names** command performs the following steps to modify the name:

1. Truncation. For example, if the maximum name length is 16, the name THIS_NAME_IS_TOO_LONG_TO_BE_ACCEPTABLE is truncated to THIS_NAME_IS_TOO.

2. Truncation and appending an index. If truncation of names results in duplicate names, these names are truncated further and appended with a number. For example, if multiple names are truncated to make the name THIS_NAME_IS_TOO, the tool renames the objects THIS_NAME_IS_T1, THIS_NAME_IS_T2, and so on up to THIS_NAME_IS_T99.

3. Renaming using a prefix and index. When Step 1 and Step 2 fail to produce unique names, the tool creates an entirely new name in the form of a prefix appended with a numeric index. By default, the prefix is "P" for ports, "U" for cells, and "N" for nets. For example, nets are named N1, N2, and so on. You can specify the prefix by using the **-prefix** option.

If all three steps fail to generate a unique name, the original name is retained and a warning message is issued.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

Irregular Bus Support

There are three modes of applying rules in the contest of handling irregular buses.

1. Rule **-flatten_port_bus** is enabled.

Ensure that all pins of a bused port have names of the form "*bus_name*[*i*]". This will typically rename buses whose HDL type involved VHDL records, SystemVerilog structs, or multidimensional arrays. Regular pin names permit write_verilog to netlist the bus as a one-dimensional bus port.

2. Rule **remove_irregular_port_bus** is enabled.

Bit-blasts any port bus that contains irregular members. The names of the bus pins would be retained in the resulting netlist.

3. Rule **flatten_multi_dimension_buses** is enabled.

Flattens multi-dimension ports, net buses, and arrays so writing out is easier. If only **-flatten_multi_dimension_buses** is turned on, then, **-remove_irregular_port_bus** would be enabled as well, to allow write_verilog to successfully write out single dimension buses with non-numeric indices.

End users could choose any of the above described modes.

EXAMPLES

The following commands change all occurrences "a" to "A" and "i" to "I" in port, cell, and net names.

```
prompt> define_name_rules UPPER_AI -map {{"a","A"},{"i","I"}}
1
prompt> report_name_rules      ; Report all defined name rules
...
Rules Name: UPPER_AI
   ...
   Report on user-defined UPPER_AI rule set
   ...
Rules Name: verilog
   ...
   Report on predefined verilog rule set
   ...
1
prompt> report_names -rules UPPER_AI  ; Report effect on names in design
                                      ; without actually changing them
```

```
  Design         Type    Object      New Name
  ------------------------------------------------------------
  leon3mp        port    reseta      resetA
  leon3mp        port    clk         clk
  ...            ...     ...         ...
```

```
prompt> change_names -rules UPPER_AI  ; Change names in design
 Information: Using name rules 'UPPER_AI'.
 Information: 908 objects (124 ports, ...) changed in design 'leon3mp'.
1
```

SEE ALSO

change_names(2)
report_names(2)
report_name_rules(2)

define_power_model

Defines a power model which describes the power intent of a reference library cell.

SYNTAX

string **define_power_model**
power_model_name
[-for *library_cells*]
{ UPF_commands }

Data Types

power_model_name	string
library_cells	list
UPF_commands	command list

ARGUMENTS

power_model_name

Specifies the name of the power model to be defined. The name should be a simple (non-hierarchical) name.

The name is scope-based, i.e., the power model cannot be defined if there is another power model with the same name in the scope where the **define_power_mode** command is executed.

-for *library_cells*

Specifies a collection of library cells that this power model can be applied to.

If this option is not specified, the power model can then be applied to any library cell.

{ UPF_commands }

UPF commands enclosed by braces will define the power intent of this power model. When the **apply_power_model** command is executed, these UPF commands will loaded in the scope of the target cells. Please see **apply_power_model(2)** for more details.

When the **define_power_model** command is executed, syntax legality of **UPF_commands** is not checked until the **apply_power_model** command is executed.

DESCRIPTION

The define_power_model command defines a power model containing other UPF commands contained within curly braces. A power model is used to define the power intent of a model and shall be used in conjunction with one or more model representations. The opening and closing curly braces encapsulating the UPF commands that describe the power model must be present in the same UPF file.

A power model can be referenced by its simple name from anywhere in a power intent description. It shall be an error to have two power models with the same name.

A power model can be applied to specific instances using **apply_power_model**.

One power model applied to a given instance may apply another power model to a descendant instance.

A power model that is not referenced by an **apply_power_model** command does not have any impact on the power intent of the design.

The **define_power_model** command provides a protected environment to the power model. Any Tcl variable used in the power model is not affected by the global variables defined outside the **define_power_model** command. Also, any Tcl variable defined inside the power model will not affect the environment outside the power model.

A power model can be parameterized by creating parameters using the **add_parameter** command. These parameters are represented as special Tcl variables which can be referenced within the power model. The parameters can be overridden with **-parameters** option of the **apply_power_model** command.

A power model can have nested hierarchy of other power models using the **apply_power_model** commands but it cannot have a nested definitions of power models, i.e. a nested **define_power_model** command.

EXAMPLES

The following example creates a power model for two library cells. Notice that there is a parameter for the top power domain name in the power model.

```
prompt> define_power_model MODEL -for {lib_cell_A lib_cell_B} {
  add_parameter DOMAIN -default domain -description "top power domain in model"
  create_supply_net VDD
  create_supply_net VSS
  create_supply_set SS -function {power VDD} -function {ground VSS}
  create_power_domain PD_$DOMAIN -supply {primary SS} -include_scope
}
```

SEE ALSO

apply_power_model(2)
report_power_model(2)
add_parameter(2)
mv.upf.power_model_search_path(3)
mv.upf.power_model_library(3)
mv.upf.power_model_verbse(3)

define_proc_attributes

Defines attributes of a Tcl procedure, including an information string for help, a command group, a set of argument descriptions for help, and so on. The command returns the empty string.

SYNTAX

string **define_proc_attributes**
proc_name
[-info *info_text*]
[-define_args *arg_defs*]
[-define_arg_groups *group_defs*]
[-command_group *group_name*]
[-return *type_name*]
[-hide_body]
[-hidden]
[-dont_abbrev]
[-permanent]

Data Types

proc_name string
info_text string
arg_defs list
group_defs list
group_name string
type_name string

ARGUMENTS

proc_name

Specifies the name of the existing procedure.

-info *info_text*

Provides a help string for the procedure. This is printed by the **help** command when you request help for the procedure. If you do not specify *info_text*, the default is "Procedure".

-define_args *arg_defs*

Defines each possible procedure argument for use with **help -verbose**. This is a list of lists where each list element defines one argument.

-define_arg_groups *group_defs*

Defines argument checking groups. These groups are checked for you in parse_proc_arguments. each list element defines one group

-command_group ***group_name***

Defines the command group for the procedure. By default, procedures are placed in the "Procedures" command group.

-return ***type_name***

Specifies the type of value returned by this proc. Any value may be specified. Some applications use this information for automatically generated dialogs etc. Please see the type_name option attribute described below.

-hide_body

Hides the body of the procedure from **info body**.

-hidden

Hides the procedure from **help** and **info proc**.

-dont_abbrev

Specifies that the procedure can never be abbreviated. By default, procedures can be abbreviated, subject to the value of the **sh_command_abbrev_mode** variable.

-permanent

Defines the procedure as permanent. You cannot modify permanent procedures in any way, so use this option carefully.

-deprecated

Defines the procedure as deprecated i.e. it should no longer be called but is still available.

-obsolete

Defines the procedure as obsolete i.e. it used to exist but can no longer be called.

DESCRIPTION

The **define_proc_attributes** command associates attributes with a Tcl procedure. These attributes are used to define help for the procedure, locate it in a particular command group, and protect it.

When a procedure is created with the **proc** command, it is placed in the Procedures command group. There is no help text for its arguments. You can view the body of the procedure with **info body**, and you can modify the procedure and its attributes. The **define_proc_attributes** command allows you to change these aspects of a procedure.

Note that the arguments to Tcl procedures are all named, positional arguments. They can be programmed with default values, and there can be optional arguments by using the special argument name *args*. The **define_proc_attributes** command does not relate the information that you enter for argument definitions with **-define_args** to the actual argument names. If you are describing anything other than positional arguments, it is expected that you are also using **parse_proc_arguments** to validate and extract your arguments.

The *info_text* is displayed when you use the **help** command on the procedure.

Use **-define_args** to define help text and constraints for individual arguments. This makes the help text for the procedure look like the help for an application command. The value for **-define_args** is a list of lists. Each element has the following format:

arg_name option_help value_help data_type attributes

The elements specify the following information:

- *arg_name* is the name of the argument.
- *option_help* is a short description of the argument.
- *value_help* is the argument name for positional arguments, or a one word description for dash options. It has no meaning for a Boolean option.
- *data_type* is optional and is used for option validation. The *data_type* can be any of: string, list, boolean, int, float, or one_of_string. The default is string.
- *attributes* is optional and is used for option validation. The *attributes* is a list that can have any of the following entries:
  - "required" - This argument must be specified. This attribute is mutually exclusive with optional.
  - "optional" - Specifying this argument is optional. This attribute is mutually exclusive with "required."
  - "value_help" - Indicates that the valid values for a one_of_string argument should be listed whenever argument help is shown.
  - "values {<list of allowable values>}" - If the argument type is one_of_string, you must specify the "values" attribute.
  - "type_name <name>" - Give a descriptive name to the type that this argument supports. Some applications may use this information to provide features for automatically generated dialogs, etc. Please see product documentation for details. This attribute is not supported on boolean options.
  - "merge_duplicates" - When this option appears more than once in a command, its values are concatenated into a list of values. The default behavior is that the right-most value for the option specified is used.
  - "remainder" - Specifies that any additional positional arguments should be returned in this option. This option is only valid for string option types, and by default the option is optional. You can require at least one item to be specified by also including the required option.
  - "deprecated" - Specifying this option is deprecated i.e. it should no longer be used but is still available. A warning will be output if this option is specified. This attribute cannot be combined with obsolete or required.
  - "obsolete" - Specifying this option is obsolete i.e. it used to exist but can no longer be used. A warning will be output if this option is specified. This attribute cannot be combined with deprecated or required.
  - "min_value" <value> - Specify the minimum value for this option. This attribute is only valid for integer and float types.
  - "max_value" <value> - Specify the maximum value for this option. This attribute is only valid for integer and float types.
  - "default" <value> - Specify the default value for this option. This attribute is only valid for string, integer and float option types. If the user does not specify this option when invoking the command this default value will be automatically passed to the associated tcl procedure.

  The default for *attributes* is "required."

Use the **-define_arg_groups** to define argument checking groups. The format of this option is a list where each element in the list defines an option group. Each element has the following format:

{<type> {<opt1> <opt2> ...} [<attributes>]}

The types of groups are

- *"exclusive"* Only one option in an exclusive group is allowed. All the options in the group must have the same required/optional status. This group can contain any number of options.
- *"together"* If the first option in the group is specified then the second argument is also required. This type of group can contain at most two options.

- \fi"related" These options are related to each other. An automatic dialog builder for this command may try to group these options together.

The supported attributes are:

- *{"label" <text>}* An optional label text to identify this group. The label may be used by an application to automatically build a grouping in a generated dialog.
- *"bidirectional"* This is only valid for a together group. It means that both options must be specified together.

Change the command group of the procedure using the **-command_group** command. Protect the contents of the procedure from being viewed by using **-hide_body**. Prevent further modifications to the procedure by using **-permanent**. Prevent abbreviation of the procedure by using **-dont_abbrev**.

EXAMPLES

The following procedure adds two numbers together and returns the sum. For demonstration purposes, unused arguments are defined.

```
prompt> proc plus {a b} { return [expr $a + $b]}

prompt> define_proc_attributes plus -info "Add two numbers" \
? -define_args {
  {a "first addend" a string required}
  {b "second addend" b string required}
  {"-verbose" "issue a message" "" boolean optional}}

prompt> help -verbose plus
Usage: plus    # Add two numbers
   [-verbose]       (issue a message)
   a              (first addend)
   b              (second addend)

prompt> plus 5 6
11
```

In the following example, the argHandler procedure accepts an optional argument of each type supported by **define_proc_attributes**, then displays the options and values received. Note the only one of -Int, -Float, or -Bool may be specified to the command:

```
proc argHandler {args} {
  parse_proc_arguments -args $args results
  foreach argname [array names results] {
    echo $argname = $results($argname)
  }
}

define_proc_attributes argHandler \
  -info "Arguments processor" \
  -define_args {
    {-Oos "oos help"      AnOos   one_of_string
      {required value_help {values {a b}}}}
    {-Int "int help"      AnInt   int     optional}
    {-Float "float help"  AFloat  float   optional}
```

```
    {-Bool "bool help"     ""      boolean optional}
    {-String "string help" AString string  optional}
    {-List "list help"     AList   list    optional}
    {-IDup "int dup help"  AIDup   int     {optional merge_duplicates}}
  } \
  -define_arg_groups {
    {exclusive {-Int -Float -Bool}}
  }
```

SEE ALSO

help(2)
info(2)
parse_proc_arguments(2)
proc(2)
sh_command_abbrev_mode(3)

define_scaling_lib_group

Associates a set of related panes to form a scaling group.

SYNTAX

string **define_scaling_lib_group**
[-name *group_name*]
[-details *detail_list*]
[-reflib *lib*]
contributing_db_libraries

Data Types

group_name	string
detail_list	list
lib	collection
contributing_db_libraries	list

ARGUMENTS

-name *group_name*

Specifies the name for the group.

By default, the command assigns a name using the following naming convention: slg<*n*>, where *n* is a unique integer.

-details *detail_list*

Displays additional information when generating the scaling groups.

Valid values are

- **basic**

  Displays the panes that match the specified logic libraries; information about the power and ground rails, including constant and dependent rails; a summary of the unscalable objects; and detailed information about each pane, including voltage, temperature, and associated logic library.

- **unscalable_objects**

  Displays the panes that match the specified logic libraries; information about the power and ground rails, including constant and dependent rails; detailed and summary information about the unscalable objects; and detailed information about each pane, including voltage, temperature, and associated logic library.

- **no_unscalable_summary**

Displays the panes that match the specified logic libraries; information about the power and ground rails, including constant and dependent rails; and detailed information about each pane, including voltage, temperature, and associated logic library. No information is provided about the unscalable objects.

- **all**

  Displays the panes that match the specified logic libraries; information about the power and ground rails, including constant and dependent rails; detailed and summary information about the unscalable objects; and detailed information about each pane, including voltage, temperature, and associated logic library.

If you do not specify this option, the command reports only a summary of the unscalable objects found in the specified logic libraries.

-reflib ***lib***

Specifies the reference library in which to search for panes.

By default, the command searches all reference libraries.

contributing_db_libraries

Specifies the logic libraries used to find the panes to group. Note that a logic library (pane) can be a part of multiple overlapping scaling groups.

This is a required option.

DESCRIPTION

The **define_scaling_lib_group** command uses the specified logic libraries to try to build a scaling group that allows interpolation in the voltage and temperature dimensions.

The command searches the reference libraries in memory for panes that match the specified logic libraries. The command assumes that the logic libraries were used to build a reference library that has been loaded into memory, and all of the logic libraries are in the same reference library.

Each logic library must be at a different characterization point and infer a single pane. Each pane must have the same process label and process number. Each reference library must have matching values for the nine library threshold values. The reference libraries must also cover all of the needed PVTs to form a full grid. For example, if you have two values for rail1, two for rail2, and two for temperature, then either contributing logic libraries are needed. If these requirements are not met, the command cannot create the group.

Given a set of panes that meet the basic requirements, determination of a valid group starts with rails and temperature. A *constant* rail or temperature (where the value is the same across all panes) is eliminated from contention. In the case of *dependent* rails, where rail1=rail2 in each of the contributing panes, one of the two is eliminated. For as many dimensions (rails or temperature) that remain, the required PVT values are listed, and then the command verifies that each required PVT is present.

The final step of creating a group is to verify which objects are scalable and which are not. To be scalable, the object must exist in every contributing pane, and each table must be a function of the same variables. Unscalable objects are not a problem. When doing a calculation, in these cases, the application falls back to the nearest matching pane.

A successful scaling group is represented as a new pane, which can be seen with the **report_lib** command. These panes are added at the end for a good reason. If your corner specification is an exact match for a basic pane in the reference library, the application uses the simpler pane that requires a single interpolation in the basic table variables rather than multiple interpolations.

A scaling group is not particularly expensive to create. In terms of memory, the cost is about a byte per object. For example, to create a scaling group for a large library with a few thousand cells, tens of thousands of arcs, dynamic power records, and so on might cost

about 10MB. The most expensive part of creating the group is looking for unscalable objects. That might take a minute or two for a large library.

EXAMPLES

In the following example, a group is successfully created.

```
prompt> set dbs \
  {wc_v0p99_t0.db wc_v0p99_t-40.db wc_v0p81_t-40.db wc_v0p81_t0}
prompt> define_scaling_lib_group $dbs -details basic -name G1
 ...'wc_v0p99_t0.db' matched pane 0 in lib WC
 ...'wc_v0p99_t-40.db' matched pane 2 in lib WC
 ...'wc_v0p81_t-40.db' matched pane 3 in lib WC
 ...'wc_v0p81_t0.db' matched pane 1 in lib WC

 ...Rail 1 -  power (vdd)
 ...Rail 2 -  pwell (gnds)
 ...Rail 3 -  nwell (vdds)

* Removing constant rail 'gnds' from consideration
* Rails vdd/vdds are dependent, removing vdds from consideration
Information: For rail 'vdd', found values: 0.81, 0.99 (SLG-024)
Information: For temperature, found values: -40, 0 (SLG-024)

 ... checking for missing objects and incompatible tables...

Reference lib: WC
 pane 0: vdd=0.99; t=0    ; file wc_v0p99_t0.db
 pane 1: vdd=0.81; t=0    ; file wc_v0p81_t0.db
 pane 2: vdd=0.99; t=-40  ; file wc_v0p99_t-40.db
 pane 3: vdd=0.81; t=-40  ; file wc_v0p81_t-40.db

 ...created scaling lib group G1
G1
```

In the following example, the specified logic libraries do not form a complete grid, so the scaling group cannot be created.

```
prompt> set dbs {wc_v0p99_t0.db wc_v0p99_t-40.db wc_v0p81_t-40.db}
prompt> define_scaling_lib_group $dbs -details basic -name G2
 ...'wc_v0p99_t0.db' matched pane 0 in lib WC
 ...'wc_v0p99_t-40.db' matched pane 2 in lib WC
 ...'wc_v0p81_t-40.db' matched pane 3 in lib WC

 ...Rail 1 -  power (vdd)
 ...Rail 2 -  pwell (gnds)
 ...Rail 3 -  nwell (vdds)

* Removing constant rail 'gnds' from consideration
* Rails vdd/vdds are dependent, removing vdds from consideration
Information: For rail 'vdd', found values: 0.81, 0.99 (SLG-024)
Information: For temperature, found values: -40, 0 (SLG-024)
Error: Missing characterization point: vdd=0.81 temp=0 (SLG-023)
```

SEE ALSO

report_lib(2)

define_test_mode

Defines a test mode to be created during DFT synthesis.

SYNTAX

status **define_test_mode**
mode_name
-usage *mode_purpose*
[-encoding {*port_name* 0 | 1, ...}]
[-target *core_mode_pair_list*]
[-transparent_mode_of *parent_mode_name*]

Data Types

mode_name string
mode_purpose string
port_name string
core_mode_pair_list list
parent_mode_name string

ARGUMENTS

mode_name

Specifies a user-defined text string containing only alphanumeric characters (a-z, A-Z, 0-9) and the underscore (_).

-usage *mode_purpose*

Specifies the mode type for a test mode. The supported mode types are **scan_compression**, **scan**, **wrp_if**, **wrp_of**, **wrp_safe**, and **logicbist**.

- **scan** is the basic scan mode operation. The scan chains are driven directly by top-level scan-in ports and they in turn drive top-level scan-out ports. This mode is used for testing all logic internal to the core.
- **scan_compression** is the DFTMAX compressed scan mode of operation. In this mode, the compressed scan chains are driven by the load compressors and the scan chains drive the unload compressors. This mode is used for testing all logic internal to the core.

  For backward compatibility, you can also use this usage to define DFTMAX Ultra compression modes if the design does not also contain DFTMAX compression modes.
- **streaming_compression** is the DFTMAX Ultra compressed scan mode of operation. In this mode, the compressed scan chains are driven by the streaming load compressors and the scan chains drive the streaming unload compressors. This mode is used for testing all logic internal to the core.
- **wrp_if** is the inward facing, or **INTEST** mode of wrapper operation. This mode is used for testing all logic internal to the core.

In this mode, wrappers are enabled and configured to drive and capture data within the design in conjunction with the internal scan chains.

- **wrp_of** is the outward facing, or **EXTEST** mode of wrapper operation. This mode is used for testing all logic external to the design. Wrappers are enabled and configured to drive and capture data outside of the design. In this mode the internal chains are disabled.
- **wrp_safe** is the safe wrapper mode. In this mode the internal chains are disabled and the internal core is protected from toggle activity. This mode is optional and provides isolation of the core while other cores are being tested. When active, safe mode enables driving steady states into or out of the design.
- **logicbist** is the LogicBIST self-test compression mode, which implements a digital logic built-in self-test capability. In this mode, the compressed scan chains are driven by a pseudo-random pattern generator for a fixed number of patterns. A multiple-input signature register (MISR) captures the resulting scan data, then compares it against an expected signature value to generate a PASS/FAIL result.

-encoding {*port_name* 0 | 1, ...}

Specifies a list of ports and the binary values that select a particular test mode. The encoding argument consists of a port name and a binary value pair that specify the test-mode port and the bit values to be used in a particular test mode. The encoding port pairs for a test mode must contain all pairs to be used in the design. See the EXAMPLES section for more information.

Before a port can be used for a test-mode port, it must be defined as a test-mode port using the **set_dft_signal** command:

set_dft_signal -type TestMode

-target *core_mode_pairs*

Specifies test-mode assignments to use for DFT-inserted cores.

This option overrides the default test-mode assignment for DFT-inserted cores. Specify a list of core and test-mode pairs to use for the top-level test mode being defined; each pair consists of a core instance name and a core test-mode name separated by a colon (:). For designs with scan compression, you can also include the name of the current design, without a colon or test-mode name, to specify that the top-level logic should be active and tested.

If a core is targeted in some test modes but not others, it is inactive in the test modes where it is not targeted. This is known as sparse targeting. (To completely exclude a core from all top-level test modes, use the **-exclude_elements** option of the **set_scan_configuration command**.)

Untargeted cores (not included in any target list across all test modes) are tested in top-level modes where the top-level logic is tested.

-transparent_mode_of *parent_mode_name*

Specifies the parent mode for the transparent test mode being defined.

A transparent mode is derived from an inward-facing parent test mode of a core-wrapped design. It is identical to its parent mode, except that

- Wrapper chains are treated as regular scan chains.
- Dedicated wrapper cells are logically transparent, although their flip-flops remain in the wrapper chains and act as observe test points on I/O paths.
- Any scan compression codecs from the referenced inward-facing mode are reused.
- Feedthrough chains drive the outward-facing wrapper scan I/Os in transparent mode.

A transparent mode allows a core-wrapped design to be tested top-down flat from a higher hierarchy level.

A transparent mode inherits all DFT configuration information from its parent mode. Do not apply any other DFT configuration

commands specifically to a transparent test mode; they are ignored. Do not specify a base mode for a transparent compressed scan mode.

DESCRIPTION

The **define_test_mode** command allows you to define a test mode for DFT synthesis. Inference for existing scan multimode designs is not supported at this time.

The information associated with a *spec* mode is interpreted as a specification of a test mode to be synthesized. This information is used during the DFT architect and insertion process. The specification information for a mode consists of protocols, port attributes, and the configuration information specific to the DFT to be inserted.

Perform multimode specifications in the following order:

1. Define TestMode signals with **set_dft_signal -test_mode all**.
2. Define all test modes, their usage, and their encoding with **define_test_mode**.
3. Define clocks, asynchronous signals, and constants common to all test modes with **set_dft_signal -test_mode all**.
4. Define mode-specific signals with **set_dft_signal -test_mode test_mode_name**.
5. Specify scan paths to be used in all test modes with **set_scan_path -test_mode all**.
6. Specify scan paths to be used in specific test modes with **set_scan_path -test_mode test_mode_name**.

Note that when the **define_test_mode** command is used, it will set the current_test_mode. Then if any of the commands listed below are used without the **-test_mode** option, the spec will not be applied to all test modes. Instead, the spec will be applied only to the mode that was last defined with **define_test_mode**. The affected commands are

set_scan_configuration

set_scan_path

set_dft_signal

set_scan_compression_configuration

set_logicbist_compression_configuration

Be sure to use the **-test_mode** option with all of these commands once the **define_test_mode command has been issued. If the spec is to be applied to all modes use -test_mode all. If the spec is to be applied to a specific test mode, use -test_mode test_mode_name.**

EXAMPLES

The following example defines three test modes in a DFTMAX compressed scan flow. The modes are ScanCompression_mode, Internal_scan1, and Internal_scan2. The encoding values are also specified.

```
prompt> set_dft_signal -view spec -type TestMode \
    -port [list i_trdy_de i_trdy_dd i_cs]
```

```
prompt> define_test_mode ScanCompression_mode \
    -usage scan_compression \
    -encoding {i_trdy_de 0 i_trdy_dd 0 i_cs 1}

prompt> define_test_mode Internal_scan1 -usage scan \
    -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 0}

prompt> define_test_mode Internal_scan2 -usage scan \
    -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 1}
```

The following example defines 9 wrapper modes:

```
prompt> set_dft_signal -view spec -type TestMode \
    -port [list i_trdy_de i_trdy_dd i_cs]

prompt> define_test_mode burn_in -usage scan  \
    -encoding {i_trdy_de 0 i_trdy_dd 0 i_cs 0 i_wr 1}

prompt> define_test_mode domain -usage scan \
    encoding {i_trdy_de 0 i_trdy_dd 0 i_cs 1 i_wr 0}

prompt> define_test_mode my_wrp_if -usage wrp_if \
    encoding {i_trdy_de 0 i_trdy_dd 0 i_cs 1 i_wr 1}

prompt> define_test_mode my_wrp_if_delay -usage wrp_if \
    -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 0 i_wr 0}

prompt> define_test_mode my_wrp_if_scl_delay -usage wrp_if \
    -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 0 i_wr 1}

prompt> define_test_mode my_wrp_of -usage wrp_of \
    -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 1 i_wr 0}

prompt> define_test_mode my_wrp_of_delay -usage wrp_of \
    -encoding {i_trdy_de 0 i_trdy_dd 1 i_cs 1 i_wr 1}

prompt> define_test_mode my_wrp_of_scl_delay -usage wrp_of \
    -encoding {i_trdy_de 1 i_trdy_dd 0 i_cs 0 i_wr 0}

prompt> define_test_mode my_wrp_safe -usage wrp_safe \
    -encoding {i_trdy_de 1 i_trdy_dd 0 i_cs 0 i_wr 1}
```

SEE ALSO

current_test_mode(2)
list_test_modes(2)
preview_dft(2)
set_dft_signal(2)
set_scan_compression_configuration(2)
set_scan_configuration(2)
set_scan_path(2)
set_streaming_compression_configuration(2)

define_user_attribute

Defines a new user-defined attribute.

SYNTAX

status **define_user_attribute**
-type string | int | float | double | boolean
-classes *class_list*
[-range_min *min*]
[-range_max *max*]
[-one_of *values*]
[-non_persistent]
[-quiet]
-name *attribute_name*
pos_attribute_name

Data Types

class_list list
min double
max double
values list
attribute_name string
pos_attribute_name string

ARGUMENTS

-type string | int | float | double | boolean

Specifies the data type of the attribute.

-classes *class_list*

Specifies the classes to which the new user-defined attribute applies. To see the valid classes, use the following command:

define_user_attribute -help

-range_min *min*

Specifies the minimum value for numeric ranges. If you specify a minimum constraint without a maximum constraint, the tool creates an attribute that accepts a value greater than or equal to *min*.

This option is valid only when the data type of the attribute is int, float, or double.

-range_max *max*

Specifies the maximum value for numeric ranges. If you specify a maximum constraint without a minimum constraint, the tool creates an attribute that accepts a value less than or equal to *max*.

This option is valid only when the data type of the attribute is int, float, or double.

-one_of *values*

Specifies a list of allowable strings.

This option is valid only when the data type of the attribute is string.

-non_persistent

Specifies that the attribute is not saved in the design library, and therefore is not persistent.

By default, user-defined attributes are saved in the design library.

-quiet

Disables warning messages that would otherwise be issued if the attribute or classes are incorrect.

-name *attribute_name*

Specifies the name of the attribute.

This option is mutually exclusive with the *pos_attribute_name* argument. You must specify either this option or the *pos_attribute_name* argument.

pos_attribute_name

Specifies the name of the attribute.

This argument is provided for compatibility with other tools. It is mutually exclusive with the **-name** option. You must specify either this option or the **-name** option.

DESCRIPTION

The **define_user_attribute** command defines a new attribute.

By default, the definition for a user-defined attribute is saved in the design library and is persistent. To make a user-defined attribute non-persistent, use the **-non_persistent** option.

To list the user-defined attributes, use the **list_attributes** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example defines an attribute named attr_1 that has a data type of double with a minimum value of 2 and a maximum value of 3.2 and can be set on cells or nets.

```
prompt> define_user_attribute -classes {cell net} -type double \
```

```
  -range_max 3.2 -range_min 2 -name attr_1
1
```

The following example defines an attribute named attr_2 that has a data type of string with valid values of true or false and can be set on nets.

```
prompt> define_user_attribute -classes net -type string \
  -one_of {true false} -name attr_2
1
```

The following example shows how to list the attribute definitions using the **list_attributes** command:

```
prompt> list_attributes
****************************************
Report : List of Attribute Definitions
Version:
Date   : Thu Jul 14 14:16:40 2011
****************************************

Properties:
   A - Application-defined
   U - User-defined
   I - Importable from design/library (for user-defined)
   S - Settable
   B - Subscripted

Attribute Name   Object   Type    Properties  Constraints
-------------------------------------------------------------
attr_1          cell    double   U          2 to 3.2
attr_1          net     double   U          2 to 3.2
attr_2          net     string   U          true, false
```

SEE ALSO

get_attribute(2)
get_defined_attributes(2)
list_attributes(2)
remove_attributes(2)
report_attributes(2)
set_attribute(2)

derive_3d_interface

Creates a derived set of bump cell, TSV (Through-Silicon Via), or bond pad objects based on an existing set of bump cell, TSV, or bond pad objects. This command maintains a one-to-one correspondence between the from_object and the derived_object, while honoring user-specified x_offset and y_offset for placement of the derived object.

SYNTAX

int **derive_3d_interface**
-from *from_object_collection*
-to_object_ref *to_object_collection*
[-x_offset *x_offset*]
[-y_offset *y_offset*]
[-coord_spec {[bl|br|tl|tr|c] [bl|br|tl|tr|c]}]
[-connect true | false]
[-name_prefix *name_prefix*]
[-orientation R0 | R90 | R180 | R270 | MX | MXR90 | MY | MYR90]
[-bond_pad_width *bond_pad_width*]
[-bond_pad_height *bond_pad_height*]

Data Types

from_object_collection collection
to_object_collection collection
x_offset float
y_offset float
name_prefix string
bond_pad_width float
bond_pad_height float

ARGUMENTS

-from *from_object_collection*

Specifies the collection of from_objects which are used to derive the new objects to be created. The collection can contain any one of the following three type of objects:

a) Collection of bump cell instances, or

b) Collection of TSV cell instances, or

c) Collection of bond pad terminal objects The collection must contain at least one object. This is a required option.

-to_object_ref *to_object_collection*

Specifies a collection of exactly one object which is used as a reference object for creating the derived object. The collection also

indicates the type of object that is derived.

a) If TSVs are to be derived, the collection object is a via def object

b) If bump cells are to be derived, the collection object is a lib_cell object

c) If bond pad objects are to be derived, the collection object is a layer object This is a required option.

-x_offset *x_offset*

Specifies the horizontal spacing from each from_object to the corresponding derived object. A negative value indicates the placement of derived_object is to the left of the from_object. Unit for this specification is micron. If not specified, the default value of 0.0 micron is used.

-y_offset *y_offset*

Specifies the vertical spacing from each from_object to the corresponding derived object. A negative value indicates the placement of derived_object is to the bottom of from_object. Unit for this specification is micron. If not specified, default value of 0.0 micron is used.

-coord_spec {bl|br|tl|tr|c bl|br|tl|tr|c}

Specifies the corner or center of the from_object which is matched against the corner or center of the derived object for applying the x_offset and y_offset specification. The specified value is a list of exactly two keywords, which must be one of the following:

a) bl - indicates the bottom-left corner of the from_object or derived_object

b) br - indicates the bottom-right corner of the from_object or derived_object

c) tl - indicates the top-left corner of the from_object or derived_object

d) tr - indicates the top-right corner of the from_object or derived_object

e) c - indicates the center of the from_object or derived_object

The first keyword is used for the from_object, and second keyword is used for the derived object. This is an optional argument. If not specified, a default value of "{c c}" is assumed.

-connect true | **false**

Specifies whether the derived object is to be connected to its corresponding from_object. This is an optional argument. If not specified, a default value of true is assumed. If from_object is connected to an existing net, it uses the same net to connect to the derived object. Otherwise, it creates a new net and connects both from_object and derived_object to this net. The newly created net is named "<from_object_name>_net". If a net with this name already exists, the command adds a unique suffix to generate a unique net name.

-name_prefix *name_prefix*

Specifies the prefix to be used for naming derived objects. This is an optional argument. If not specified, the command uses the following specification for determining the name prefix.

a) default_tsv_array<uniqueId>_<viadef> for derived TSVs

b) default_bondpad_array<uniqueId>_<layer_name> for derived Bond Pads

c) default_bump_array_<uniqueId> for derived Bumps

The full name of derived object depends on how the from_objects are named. If all from_objects have names of the form: "<from_prefix>_index1_index2", then the derived objects are named "<name_prefix>_index1_index2". Otherwise, derived objects are named "<name_prefix>_<from_object_name>".

-orientation R0 | **R90** | **R180** | **R270** | **MY** | **MXR90** | **MX** | **MYR90**

Specifies the orientation of the derived objects to be created. If this option is not specified, the default orientation is R0.

-bond_pad_width *bond_pad_width*

Specifies the width of Bond Pad shape created on specified bond pad layer. If the "-to_object_ref" argument specifies a bond pad layer name, this argument is mandatory, and it must specify a positive value for derived bond pad shape width. The value is specified in microns. If the derived object is not of bond pad type, this argument is ignored.

-bond_pad_height *bond_pad_height*

Specifies the height of bond pad shape created on a specified bond pad layer. If the "-to_object_ref" argument specifies a bond pad layer name, this argument is mandatory, and it must specify a positive value for derived bond pad shape height. Value is specified in microns. If derived object is not of bond pad type, this argument is ignored.

DESCRIPTION

This command creates a set of derived bump cell, TSV, bond pad objects based on an existing set of bump cell, TSV, or bond pad objects. The command maintains a one-to-one correspondence between the from_object and the derived_object, while honoring user-specified x_offset and y_offset for placement of the derived object.

EXAMPLES

The following example creates a set of TSV objects based on existing bump cell objects.

```
set bumpcell_coll   [get_cells bump_array_name*]
set tsv_viadef_coll [get_via_defs -tech [get_techs *] VIAB1]
derive_3d_interface -from $bumpcell_coll -to_object_ref $tsv_viadef_coll  \
  -x_offset 10.0 -y_offset -10.0 -coord_spec {br tl}
```

The following example creates a set of bond pad objects based on existing TSV objects. The center of the bond pad is aligned with the center of corresponding TSV instance.

```
set tsvcell_coll      [get_cells THROUGH* -within { {8999 1999} {10000 3000} }]
set bondpad_layer_coll [get_layers AP]
derive_3d_interface -from $tsvcell_coll -to_object_ref $bondpad_layer_coll \
  -bond_pad_width 0.8 -bond_pad_height 0.5
```

The following example creates a set of bump cell objects based on existing Bond pad objects. In this example, the bump cell instances are not connected to the corresponding Bond pad terminals.

```
set bond_pad_coll     [get_terminals bond_pad_AP_0_*]
set bump_lib_cell_coll [get_lib_cells */BUMP]
derive_3d_interface -from $bond_pad_coll -to_object_ref $bump_lib_cell_coll \
  -x_offset 10.0 -name_prefix xyzPAD22 -connect false
```

SEE ALSO

create_tsv_array(2)

create_bump_array(2)
create_bond_pad_array(2)

derive_cell_snap_data

Constructs data used for snapping multi height standard cells.

SYNTAX

status **derive_cell_snap_data**
[-force]

ARGUMENTS

-force

Forces the cell snapping data to rebuild, even if the data is already constructed. If this option is not specified the command will not rebuild existing snapping data.

DESCRIPTION

The **derive_cell_snap_data** command derives a snapping data used to snap standard cells to placement sites when assigned site definitions does not match the cell size.

EXAMPLES

The following command derives data used for snapping cells into placement sites.

prompt> **derive_cell_snap_data**

SEE ALSO

snap_objects(2)
move_objects(2)
rotate_objects(2)
copy_objects(2)
set_snap_setting(2)
get_snap_setting(2)

derive_clock_balance_constraints

Generate clock balance groups based on cross clock timing relations.

SYNTAX

string **derive_clock_balance_constraints**
[-slack_less_than]

ARGUMENTS

-slack_less_than

Specify the threshold for interclock timing slack to consider.

DESCRIPTION

This command collects a set of clocks which have timing slack smaller than the threshold

Multicorner-Multimode Support

EXAMPLES

prompt> **derive_clock_balance_constraints -slack_less_than 0.1**

SEE ALSO

create_clock_balance_group(2)
report_clock_balance_groups(2)
remove_clock_balance_groups(2)
balance_clock_groups(2)

derive_clock_balance_points

Derive set_clock_balance_points constraints at the clock sinks from ideal SDC clock latencies. The clock balance point constraints will instruct CTS to maintain the clock arrival time differences between pairs of sinks when switching from ideal to propagated clocks.

SYNTAX

status **derive_clock_balance_points**
[-clocks *clock_list]*
[-corners *corner_list*]
[-reference_latency *latency*]
[-output *file*]

Data Types

clock_list collection
corner_list collection
latency float
file string

ARGUMENTS

-clocks *clock_list*

Balance point constraints will be generated for the sinks of these clocks. If this options is not specified, balance points will be created for all ideal primary clocks of all active scenarios that have hold and/or setup analysis enabled.

-corners *corner_list*

Only generate balance point constraints for these corners. Only scenarios with the clock's mode and one of the specified corners will be considered.

-reference_latency *latency*

Balance point constraints will reflect the difference between ideal clock latencies at sinks and this reference latency at the sinks.

-output *file*

Instead of applying the clock balance points to the design, write the balance point constraints to the named file. The balance points are written out in Tcl format. This allows for inspection and modification of the balance points.

DESCRIPTION

The **derive_clock_balance_points** command will create *set_clock_balance_point* constraints based on the ideal latencies as specified by the SDC *set_clock_latency* command. The clocks to be processed should be primary clocks in ideal mode. Generated clocks or clocks that are in propagated mode are not supported. The command will convert the ideal pre-CTS latency modeling of the clock sinks to CTS goals. The command uses a reference latency per clock. The default reference latency is the sum of the source- and network-latency as specified for the clock. The source- and network latency can be specified using SDC's *set_clock_latency* on the clock objects. In order to derive the balance point delay value, the ideal mode total clock latency at a sink is subtracted from the reference latency. The result of the subtraction is applied as the *-delay* value of *set_clock_balance_point*. Each created set_clock_balance_point constraint is clock- and corner-specific. If the resulting delay value is 0.0, then the balance point constraint will only be created if there is already an existing balance point constraint active for that (pin,clock,corner). Note that the existing balance point constraint may be clock-independent or corner-independent. A user-specified reference latency will override the default reference latency.

In a standard situation where only set_clock_latency assertions are defined at the clock objects, the default reference latency will match the ideal total latency at all sinks. In that case, all -delay values will be 0.0. As a consequence, no balance point constraints will be generated. If a selected number of sinks have a total ideal latency that differs from the reference latency, then only for those sinks clock balance point constraints will be generated.

Multicorner-Multimode Support

EXAMPLES

The following example creates two balance point constraints.

```
prompt> set_clock_latency -source 0.5 [get_clock clk]
prompt> set_clock_latency 1.0 [get_clock clk]
prompt> set_clock_latency 1.5 [get_pin reg1/CK]
prompt> set_clock_latency 0.5 [get_pin reg2/CK]
prompt> derive_clock_balance_points -clocks [get_clock clk]
```

Created balance points:

```
set_clock_balance_point -delay -0.5 -corner default -balance_points \
        [get_pin reg1/CK] -clock clk
set_clock_balance_point -delay 0.5 -corner default -balance_points \
        [get_pin reg2/CK] -clock clk
```

The following example uses 0.0 as the reference latency.

```
prompt> set_clock_latency -source 0.5 [get_clock clk]
prompt> set_clock_latency 1.0 [get_clock clk]
prompt> set_clock_latency 1.5 [get_pin reg1/CK]
prompt> set_clock_latency 0.5 [get_pin reg2/CK]
prompt> derive_clock_balance_points -clocks [get_clock clk] \
    -reference_latency 0.0
```

Created balance points:

```
set_clock_balance_point -delay -2.0 -corner default -balance_points \
        [get_pin reg1/CK] -clock clk
set_clock_balance_point -delay -1.0 -corner default -balance_points \
        [get_pin reg2/CK] -clock clk
set_clock_balance_point -delay -1.5 -corner default -balance_points \
        <all_sinks_except_reg1_and_reg2> -clock clk
```

SEE ALSO

derive_clock_cell_references

Generate a list of LEQ references for the clock gates and ICGs in the design and make it available for CTS.

SYNTAX

string **derive_clock_cell_references**
[-output *filename*]

Data Types

filename string

ARGUMENTS

-output *filename*

Writes the list of LEQ references for clock gates and ICGs in the output file provided.

DESCRIPTION

The command identifies the clock gates and ICGs in the design. This does not include buffers or inverters in the design. It identifies the logically equivalent references for these cells. These references are then made available to CTS for usage.

Before running the **derive_clock_cell_references** command, if references have been excluded for CTS usage these will not be considered by **derive_clock_cell_references**. Exclusion could have been made either by marking references as 'dont_use' during library preparation or by marking dont_touch on these references or by calling **set_lib_cell_purpose -exclude cts** for the concerned references.

When output option is used, the settings will be written to specified file instead of applying to the design. The file will also list of references which are disabled due to dont_touch, dont_use or user driven exclude in the comments format. You can then review this script and source it to provide the clock tree references required for the CTS engine. Generated output file will contain following sections:

- References enabled for CTS: Logical equivalent references of clock gates and ICG of current design which does not have dont_touch, dont_use or user exclude(include_purpose doesn't have CTS) for CTS."

Following options will be in comment format. User can review the tcl file and enable by uncommenting command for enabling library cell if required:

- References already available for CTS: Library cells which are already available for CTS either by user setting or library setting.

- Dont touch references: Library cell with dont_touch attribute
- Dont use references: Library cell with dont_use attribute
- User driven exclude references: User has explicitly excluded this lib cell for CTS purpose

EXAMPLES

The following example shows that all the lib_cells other than 'and02d8' and 'and02d16' are made available for CTS.

```
prompt> set_lib_cell_purpose -exclude cts [get_lib_cells {tech_lib/and02d8 tech_lib/and02d16}]
prompt> derive_clock_cell_references
```

The following example shows how user can control the references enabled for CTS as a part of **derive_clock_cell_references** command.

```
prompt> derive_clock_cell_references -output cts_leq_set.tcl
```

SEE ALSO

set_lib_cell_purpose(2)

derive_hier_antenna_property

Derives the hierarchical antenna properties of all the top-level ports in the specified cell. You should use this command for designs routed with router.

SYNTAX

status **derive_hier_antenna_property**
-design_name *design_name*

Data Types

design_name string

ARGUMENTS

-design_name *design_name*

Specifies the design name from which to derive antenna properties.

DESCRIPTION

This command derives the hierarchical antenna properties of all the top-level ports in the given design. It is recommended that you use this command for designs routed with router. The hierarchical antenna properties include gate size, routing area, and diode protection.

This command should be run within the child cell of each macro and saved. When antenna analysis is done at the top level, the properties that were previously set can be seen in each macro pin. When nets connecting to macro pins are analyzed, the antenna information from each macro pin is used for analyzing that pin. This eliminates repetitive calculation of antenna properties of each macro or flattening the design.

The hierarchical antenna properties are calculated based on the following information:

- Routing contained in the DESIGN view of the top-level cell.
- The hierarchical antenna properties of the design
- The gate sizes and the diode protection for each port of the library cells.

The hierarchical antenna properties are stored in the FRAME view of the cell. Both the DESIGN and FRAME views must exist before you execute this command.

Perquisites

The design library must be opened first.

Both the DESIGN view and the FRAME view of the cell must exist and be writable before the command is run.

The antenna properties of all port instances in the cell must be properly set. Otherwise, the antenna property calculated will not be accurate. The antenna properties can be set by:

- Running the **derive_hier_antenna_property** command on child cells, if the child cell is routed by router.
- Set the gate size and diode protection value of each port, if the child cell is a standard cell.

Limitations

The **derive_hier_antenna_property** command does not work on standard library cells or hard macros because the tool does not read the diffusion layer in the layout. The results might not be accurate if hard macros exist in the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the use of the command on a cell instance named my_cell.

```
prompt> derive_hier_antenna_property -design_name my_design
```

derive_macro_relative_location

Derives a set of relative location constraints for hard macros and writes out **set_macro_relative_location** commands to describe the constraints.

SYNTAX

status **derive_macro_relative_location**
[-cells *macro_list*]
[-offset_type { auto|fixed|scalable auto|fixed|scalable }]
[-anchor_corner bl | tl | tr | br | *corner_index*]
[-anchor_object *anchor_object*]
[-use_bbox]
[-cluster_spacing *xy_distance* | {*x_distance y_distance*}]
[-hierarchical]
[-output_file *file_name*]

Data Types

macro_list list
corner_index integer
anchor_object core_area, move_bound, voltage_area, voltage_area_shape
xy_distance float
x_distance float
y_distance float
file_name string

ARGUMENTS

-cells *macro_list*

Specifies the macros for which to derive relative location constraints. If not specified, the command derives constraints for all macros which are not marked physical-only in a hierarchical design when you specify the **-hierarchical** option. If want physical-only macros to be included as well then use -cells option to pass the macro cell list to the command.

-offset_type { auto|fixed|scalable auto|fixed|scalable }

Specifies the preference for the offset type when writing out the location constraints. Valid offset type values are: auto, fixed and scalable as follows:

- fixed: Command writes out offsets as absolute values.
- scalable: Command writes out offset values as ratios between the absolute offset values and the corresponding dimensions of the block/core_area/move_bound/voltage_area_shape's size.

- auto: Command automatically determines the offset type: fixed or scalable. This is the default.

The constraints happen in one macro cluster (For macro1 anchor to macro2, if macro1+its_keepout_margin+finfet/litho_grid_pitch touches macro2+its_keepout_margin+finfet/litho_grid_pitch, the two macros are one macro cluster) are fixed type. The constraints happen in one macro array are fixed type. For macro anchor to block/core_area corner, if macro+its_keepout_margin+finfet/litho_grid_pitch touches the block_corner-its_inner_keepout_margin/core_area_corner, the constraint's offset type is fixed. Option -cluster_spacing is used to control/set the user specified macro spacing. The hard/hard_macro/placement_blockage keepout margin and block's inner keepout margin are considered. The offset type can be specified separately on X and Y dimension.

-anchor_corner bl | tl | tr | br | *corner_index*

Restricts the eventual anchor corner on the block of all specified macros. Valid values for this option are: bl, tl, tr, br, or a positive integer no larger than the total corner count of the current design block. The corner index of a block starts from 1, corresponding to the bottom leftmost corner, and increases in the clockwise direction proceeding around the macro. If **-anchor_corner** is one of bl, tl, tr or br, the command derives constraints based on the bounding box of the current block, as if **-use_bbox** option were specified.

-anchor_object *anchor_object*

Specifies the core_area/move_bound/voltage_area/voltage_area_shape as the eventual anchor object for all specified or derived macros in it to anchor on. For move_bounds and voltage_areas, the first shape is used. If this option is not specified, the default anchor object is the hard macros' parent block.

-use_bbox

Specifies that the command uses the bounding box instead of the rectilinear boundary of the shape when deriving location constraints. If not specified, the command derives constraints based on the rectilinear boundary of the block.

-cluster_spacing

This option is used to control the macro cluster. For macro1 anchor to macro2, if macro1+its_keepout_margin+cluster_spacing touches macro2+its_keepout_margin+cluster_spacing, the two macros are in one macro cluster, and the constraint between these two macros is fixed offset type. For macro anchor to block/core_area corner, if macro+its_keepout_margin+cluster_spacing touches the block_corner-its_inner_keepout_margin/core_area_corner, the constraint's offset type is fixed. Default cluster_spacing value is finfet/litho_grid_pitch.

-hierarchical

The command derives constraints for all hard macros in hierarchical design with this option.

-output_file *file_name*

Specifies the output file name in which to write the constraints. By default, relative locations are written to the console and not saved to a file.

DESCRIPTION

This command prints out macro relative location constraints as a set of **set_macro_relative_location** commands. You can limit the output to only a specified set of macros with the **-cells** option, otherwise by default the tool writes out macro relative location constraints for all macros which are not marked as physical-only in hierarchical design with the **-hierarchical** option. If want physical-only macros to be included as well then use -cells option to pass the macro cell list to the command.

EXAMPLES

The following example writes out relative location constraints for all hard macros in the hierarchical design to the console.

```
prompt> derive_macro_relative_location -hierarchical
set_macro_relative_location -target_object [get_cells cmem0/ddata0_0/x0_t0]
 -target_orientation MX -target_corner bl -anchor_object [get_cells cmem0/idata0_1/x0_t0]
 -anchor_corner tr -offset {0.0000 10.7200}
[ more set_macro_relative_location commands printed to console ]
Derived relative location constraints for 38 macros.
1
```

The following example writes out the relative location constraints only for hard macros U1 and U2 in the current design and save results to **relative_locations.out**.

```
prompt> set macros {U1 U2}
prompt> derive_macro_relative_location -cells [get_cells $macros] \
  -output_file relative_locations.out
Derived relative location constraints for 2 macros.
1
```

SEE ALSO

create_macro_relative_location_placement(2)
create_placement(2)
remove_macro_relative_location(2)
report_macro_relative_location(2)
set_macro_relative_location(2)
write_macro_relative_location(2)

derive_mask_constraint

Derives mask constraint for a specified shapes and vias from underlying tracks.

SYNTAX

status **derive_mask_constraint**
[*object_list*]
[-derive_cut_mask]
[-follow_pin_mask]
[-follow_wider_object_mask]
[-verbose]

Data Types

object_list collection

ARGUMENTS

object_list

Specifies the collection or selection set containing objects to derive mask constraint. If this option is not specified, the global selection set is used.

-derive_cut_mask

Specified if cut mask constraints should be derived. By default, the command only derives mask constraints for shapes and via enclosures.

-follow_pin_mask

Specified if shape mask constraints should be derived from overlapping/touching pins or ports. By default the shape mask is derived from track only. This option works only for shapes and via enclosures, via cuts are not affected.

-follow_wider_object_mask

Specified if shape mask constraints should be derived from wider overlapping/touching objects. By default the shape mask is derived from track only. This option works only for shapes and via enclosures, via cuts are not affected.

-verbose

Specified if detailed messages should be print.

DESCRIPTION

This command derives the color mask for given wires and vias from the nearest underlying tracks. In case of wide wires that occupy multiple tracks, it takes the mask color opposite to next unoccupied track. The latter is guaranteed to work only for two mask scenarios.

This command ignores the global edit setting *get_edit_setting -update_color_mask*.

EXAMPLES

The following example updates color mask of net ABC shapes.

```
prompt> change_selection [get_shapes -of_object [get_net ABC]]
prompt> derive_mask_constraint
```

The alternative syntax uses collection to specify objects.

```
prompt> derive_mask_constraint [get_shapes -of_object [get_net ABC]]
```

SEE ALSO

get_selection(2)
change_selection(2)
set_edit_setting(2)
get_edit_setting(2)

derive_metal_cut_routing_guides

Creates metal cut allowed and forbidden preferred grid extension routing guides. Metal cut allowed routing guides cover the area taken up by all the placable site rows reduced by the vertical and horizontal shrink factor. The vertical shrink factor is expressed as a percentage of the smallest site row height and the default is 50%. It can be set using the chipfinishing.metal_cut_allowed_vertical_shrink_factor app option. The horizontal shrink factor is expressed as a percentage of the smallest site row width. It can be set using the chipfinishing.metal_cut_allowed_horizontal_shrink_factor app option. Finally, forbidden preferred grid extension routing guides are created to cover the remaining area up to the boundary.

SYNTAX

collection **derive_metal_cut_routing_guides**
[-site_def *site_def*]
[-boundary_cells *lib_cell_names*]
[-treat_fixed_as_blockage]
[-add_metal_cut_allowed]
[-add_standard_cell_region]
[-check_only]
[-error_view *file_name*]]

Data Types

site_def string or collection
lib_cell_names collection
file_name string
percentage float

ARGUMENTS

-site *site_def*

Specifies the site definition that the shrink factors are calculated from.

-boundary_cells *lib_cell_name*

Specifies the list of boundary cells.

-treat_fixed_as_blockage

Specifies whether existing fixed cells are treated as blockage.

-add_metal_cut_allowed

Specifies to add metal cut allowed routing guide underneath standard cell placable area.

-add_standard_cell_region

Specifies to add standard cell region routing guide underneath standard cell placable area.

-check_only

The command checks route guides with this option.

-error_view

Specifies the output file name for error browser.

-standard_cell_region_vertical_shrink_factor ***percentage***

Specifies the vertical area between place-able area for route guide creation as a percentage of the unit tile. The default value is 0.5.

-standard_cell_region_horizontal_shrink_factor ***percentage***

Specifies the horizontal area between place-able area for route guide creation as a percentage of the unit tile. The default value is 0.0.

-boundary_cells ***lib_cell_name***

Specifies the list of boundary cells.

DESCRIPTION

The **derive_metal_cut_routing_guides** command enables you to create metal cut allowed and forbidden preferred grid extension routing guides.

EXAMPLES

The following example derives routing guides using the site definition unit.

prompt> **derive_metal_cut_routing_guides -site_def unit** {abc}

SEE ALSO

create_boundary_cells(2)

derive_perimeter_constraint_objects

Generate the specified perimeter constraint objects to constraint detailed routing.

SYNTAX

int **derive_perimeter_constraint_objects**
-perimeter_objects {metal_preferred | mpndrg_nonpreferred | route_blockage_preferred | route_blockage_nonpreferred}
[-layers {*layer_name*}]
[-short_metal_length {*layer_length_pair*}]
[-metal_spacing {*layer_spacing_pair*}]
[-spacing_from_boundary {*layer_spacing_pair*}]
[-pin_cutout]
[-spacing_preferred *spacing_value*]
[-width_preferred *width_value*]
[-spacing_nonpreferred *spacing_value*]
[-width_nonpreferred *width_value*]
[-stub *stub_value*]
[-spacing_cut_layer_horizontal *spacing_value*]
[-width_cut_layer_horizontal *width_value*]
[-spacing_cut_layer_vertical *spacing_value*]
[-width_cut_layer_vertical *width_value*]
[-hierarchical]
[-check_only]
[-error_view *file_name*]]

Data Types

layer_name list
layer_length_pair list
layer_spacing_pair list
spacing_value float
width_value float
stub_value list
file_name string

ARGUMENTS

-perimeter_objects {metal_preferred | mpndrg_nonpreferred | route_blockage_preferred | route_blockage_nonpreferred}

Specifies the objects to be generated along the preferred and nonpreferred perimeter edges of a block or design. More than one type of objects can be specified. This option must be provided. The route blockage and mpndrg (max pattern nonpreferred direction route guide) objects serve certain routing objectives due to process technology requirements.

-layers {*layer_name*}

Specifies the metal routing layers or via cut layers where objects should be generated. The default value is all metal routing layers or via cut layers.

-short_metal_length {*layer_length_pair*}

Specifies the length for the short metal object at the preferred routing direction perpendicular to the edge at the non-preferred direction as {{layer_name length_value} ...}

-metal_spacing {*layer_spacing_pair*}

Specifies the spacing between the metal objects perpendicular to the edge at the non-preferred routing direction as {{layer_name spacing_value} ...}

-spacing_from_boundary {*layer_spacing_pair*}

Specifies the spacing (from the edge) for the object perpendicular to the edge at the non-preferred routing direction as {{layer_name spacing_value} ...}

-pin_cutout

Specifies that when routing blockages are generated, cutout will be applied when I/O pins are present on the corresponding edge of the blockage. The cutout allows routing access to the pins. By default, no cutout is applied.

-spacing_preferred *spacing_value*

Specifies the spacing (from the edge) for the object along an edge at the preferred routing direction. The unit is micron. All layers have the same value if specified. If this option is not specified, the minimum metal spacing of the layer from the tech file will be used. When metal_preferred is specified, if this option is not specified, the created metal runs along the closest wire track of the edge while keeping the metal shape within the boundary of the design, and if this option is specified, the created metal shape will be snapped to wire track.

-width_preferred *width_value*

Specifies the width of the object along an edge at the preferred routing direction. The unit is micron. All layers have the same value if specified. If this option is not specified, the default metal width of the layer from the tech file will be used.

-spacing_nonpreferred *spacing_value*

Specifies the spacing (from the edge) for the object along an edge at the non-preferred routing direction. The unit is micron. All layers have the same value if specified. If this option is not specified, the minimum metal spacing of the layer from the tech file will be used.

-width_nonpreferred *width_value*

Specifies the width of the object along an edge at the non-preferred routing direction. The unit is micron. All layers have the same value if specified. If this option is not specified, the default metal width of the layer from the tech file will be used.

-stub {*stub_value*}

Specifies the offset (from the center), starting position (from the edge), the length, and the spacing between adjacent stubs for metal stubs at the preferred routing direction as {offset_value start_value length_value spacing_value}. The unit is micron. All layers have the same values if specified. The created metal stubs run along the edge while keeping the preferred spacing. If this option is not specified, the metal rectangle will be created. This option can only be specified when metal_preferred is specified.

-spacing_cut_layer_horizontal *spacing_value*

Specifies the spacing (from the edge) for the object along an edge at the horizontal direction. The unit is micron. All via cut layers have the same value if specified. If this option is not specified, the max value of spacing_preferred and spacing_nonpreferred will be used.

-width_cut_layer_horizontal *width_value*

Specifies the width of the object along an edge at the horizontal direction. The unit is micron. All via cut layers have the same value if specified. If this option is not specified, the max value of width_preferred and width_nonpreferred will be used.

-spacing_cut_layer_vertical *spacing_value*

Specifies the spacing (from the edge) for the object along an edge at the vertical direction. The unit is micron. All via cut layers have the same value if specified. If this option is not specified, the max value of spacing_preferred and spacing_nonpreferred will be used.

-width_cut_layer_vertical *width_value*

Specifies the width of the object along an edge at the vertical direction. The unit is micron. All via cut layers have the same value if specified. If this option is not specified, the max value of width_preferred and width_nonpreferred will be used.

-hierarchical

The command derives perimeter constraint objects for all soft macros in hierarchical design with this option.

-check_only

The command checks route guides with this option.

-error_view

Specifies the output file name for error browser.

DESCRIPTION

Generate the specified perimeter constraint objects to constraint detailed routing. The size and locations of the objects are specified by the Techfile by default. Command options can specify parameter values that override the Techfile.

EXAMPLES

The following example creates the perimeter constraint objects for metal_preferred and mpndrg_nonpreferred.

prompt> **derive_perimeter_constraint_objects -perimeter_objects { metal_preferred mpndrg_nonpreferred } -layers { M1 M3**

The following example creates the perimeter constraint objects for metal stubs.

prompt> **derive_perimeter_constraint_objects -perimeter_objects { metal_preferred } -layers { M1 } -width_preferred 0.2 -st**

The following example creates the perimeter constraint objects for route_blockage_preferred and route_blockage_nonpreferred.

prompt> **derive_perimeter_constraint_objects -perimeter_objects { route_blockage_nonpreferred route_blockage_preferred**

SEE ALSO

derive_boundary_routing_blockages(2)
derive_metal_cut_routing_guides(2)

derive_pin_access_route_guides(2)

derive_pg_mask_constraint

Derives mask constraint for PG shapes, vias, via matrixes and shape patterns.

SYNTAX

status **derive_pg_mask_constraint**
[-nets *netname_list*]
[-overwrite]
[-derive_cut_mask]
[-always_align_color_layers *layer_name_list*]
[-verbose]
[-ignore_std_cell]
[-color_off_track_shape]
[-check_fix_shape_drc]

Data Types

netname_list list
layer_name_list list

ARGUMENTS

-nets *netname_list*

Specifies a list of PG nets on which to derive mask constraints. If not specified, mask constraints will be derived for all PG nets.

-overwrite

Specifies that the existing mask constraints of PG objects should be removed. If specified, existing mask constraints will be erased and reassigned.

-derive_cut_mask

Specifies that cut mask constraints should be derived. By default, the command only derives mask constraints for shapes and via enclosures.

-always_align_color_layers

Specifies layers on which non-default width wires have same color as aligned routing tracks. By default, the nondefault width wires have opposite color to aligned routing tracks.

-verbose

Specifies that detailed messages should be printed.

-ignore_std_cell

Specifies that DRC checking against standard cells should be ignored for cut mask constraint assignment.

-color_off_track_shape

Specifies to derive mask of shape not aligned on track. By default assigns color mask only for shapes which center is aligned with colored track. If to be colored shape overlaps with already colored shape without a conflict, color is derived from overlapping shape rather than colored track.

-check_fix_shape_drc

Specifies to check shape DPT drc and fix by rotating mask of shape when mask is derived from track color. If flipping mask doesn't resolve drc, mask will retain its color from track. This flow is recommended to use with off-track shapes, but it will impact runtime for drc checking. By default this option is off.

DESCRIPTION

This command derives the mask constraints for PG wires, vias, via matrixes and shape patterns of specified nets. For wires and via enclosures, the mask constraint assignment follows these two rules: 1) assign same mask constraint for touching shapes if possible, and 2) assign mask constraint based on the underlying routing track mask constraint. For via cuts, the command will assign mask_one or mask_two if no DRC violation is seen. For shape patterns the limitations are: Support only "uniform type", and not "non-uniform type" shape pattern coloring; Overlap with colored shapes condition is not supported now, only shapes that are centered on tracks are supported; DRC will not be checked when color mask to shape patterns is assigned..

EXAMPLES

The following example assigns mask_constraint for VDD wires, vias, via matrixes and shape patterns.

```
prompt> derive_pg_mask_constraint -nets VDD -derive_cut_mask
```

SEE ALSO

remove_pg_mask_constraints(2)
report_pg_mask_constraints(2)
set_pg_mask_constraint(2)

derive_pin_access_routing_guides

Generates a set of routing guides and routing blockages around the specified hard macros to improve routability to their pins.

SYNTAX

status **derive_pin_access_routing_guides**
-cells *cells*
-layers *layers*
-x_width *distance*
-y_width *distance*
[-boundary_threshold *distance*]
[-pin_extension *ext_value*]
[-nonpreferred_direction]

Data Types

cells collection
layers collection
distance float
ext_value float

ARGUMENTS

-cells *cells*

Specifies the hard macros for which to create the pin access routing guides. This is a required option.

-layers *list*

Specifies the layers on which to create the pin access routing guides. This is a required option.

-x_width *distance*

Specifies the x-direction width in microns of the routing guide. This is a required option.

-y_width *distance*

Specifies the y-direction width in microns of the routing guide. This is a required option.

-boundary_threshold *distance*

Specifies a distance threshold to include an internal macro pin for processing. By default, a pin not touching an edge of the macro is excluded. If the distance of an internal macro pin is within the threshold value, it will be included. The unit is micron.

-pin_extension *ext_value*

Generate pin extension on the metal routing layers. The *ext_value* specifies the protrusion of the pin extension beyond the routing guide specified by either the **-x_width** or **-y_width** options. The unit is microns. Negative values for *ext_value* are not allowed. The width of the pin extension is the same as the width of the corresponding pin. Pin extension on the non-preferred routing direction is also generated if the **-nonpreferred_direction** option is specified. No pin extension is generated if this option is omitted. Pin extension is not generated for logically unconnected pins.

-nonpreferred_direction

Specifies that the cutout of the metal layer blockage should also be done on the nonpreferred routing direction to access the macro pins. By default only the preferred routing direction pin access are allowed.

DESCRIPTION

The **derive_pin_access_routing_guides** command creates pin access routing guides for the specified hard macros. The pin access routing guides enable the router to connect to the macro pins. It is often necessary to create pin access routing guides when reusing hard macros in a smaller technology node. Use this command after placing the macros and before routing.

This command performs the following tasks:

- Creates a pin access routing guide along the boundary of the specified macros

  The routing guide enables straight routing to the pin using preferred direction routing.

- Creates a zero-spacing metal blockage on the specified layers that forms a ring around each specified macro, including its keepout margin; the metal blockage has a cutout for each pin on the layer

  The metal blockage prevents unrelated routing near the macro, while the cutouts enable routing access to the pins.

- Creates a zero-spacing via blockage on the via layers adjacent to the specified layers that covers each macro, including its keepout margin

  The via blockage prevents routing direction changes when connecting to the macro pins.

- Sets the placement status of each macro to fixed so that it cannot be moved by the automatic macro placement command

- Sets the **is_rectangle_only_rule_waived** attribute on the macro pins to **true**

  This attribute waives the rectangle-only rule to allow proper routing to pins with different sizes.

- Reports pins that violate the minimum spacing rules specified with the **minSpacingForMacroPin** attribute in the Layer section of the technology file

- Places the generated routing guides and blockages in an edit group with their associated macro so that you can move the objects together in the GUI editor

EXAMPLES

The following example creates pin access routing guides and blockages on the Metal2 and Metal 3 layers for the myMacro1 and mylp2 macros. The routing guides are 0.6um wide in the x-direction and 0.7um wide in the y-direction.

```
prompt> derive_pin_access_routing_guides \
  -cells { myMacro1 mylp2 } -layers { Metal2 Metal3 } \
  -x_width 0.6 -y_width 0.7
```

SEE ALSO

create_routing_guide(2)
route_auto(2)
route_detail(2)

derive_placement_blockages

Used to automatically create blockages and route blockages to cover thin channels, based on current placement.

SYNTAX

status **derive_placement_blockages**
[-hierarchical]
[-force]

ARGUMENTS

-hierarchical

Specifies that the command should also work on the blocks in the design. By default, it will stop at block boundaries.

-force

Specifies that the command should always create the blockages. By default, it will not create the blockages it thinks utilization might get too high after blockages are inserted.

DESCRIPTION

This command is used to create placement blockages covering thin channels between hard macros and block boundaries. This command also creates routing blockages for all user created route blockages (created using **create_keepout_margin -type route_blockage**). This is useful to set up the block for block implementation.

For the thin channel placement blockages, if the **place.floorplan.sliver_size** application option is set, then this is used as the maximum size for a thin channel. If this application option is not set, the tool estimates the size of a thin channel based on hard_macro keepout margins. These keepouts are either user created using **create_keepout_margin -type hard_macro** or auto generated. See the for **plan.place.default_keepout** man page.

To remove these derived blockages use the following command:

remove_placement_blockages [get_placement_blockages -filter is_derived==true]

EXAMPLES

The following example derives placement blockages for the top-level design.

```
prompt> derive_placement_blockages
1
```

SEE ALSO

create_keepout_margin(2)
create_placement(2)

derive_preferred_macro_locations

Extracts the current macro locations as preferred macro locations. If specified, the command writes to locations to a constraint file as **set_macro_constraints** commands. The file can be sourced in a subsequent session and used to guide the macro placement. Use this command to capture a "good" macro placement achieved in one session and use the placement in a subsequent session to reduce sensitivity to small netlist modifications and other changes.

SYNTAX

status **derive_preferred_macro_locations**
[*hard_macro_list*]
[-file *file_name*]

Data Types

hard_macro_list list
file_name string

ARGUMENTS

hard_macro_list

Specifies the list of hard macros for which preferred location constraints are calculated. When omitted, all the macros in the current design are considered.

-file *file_name*

Writes the resulting preferred location constraints to *file_name*. By default, the preferred location constraints are set in the current session.

DESCRIPTION

This command generates preferred location constraints based on the current macro locations. Use this command to write out the result to a file, or to set the constraints internally in the current session. The resulting preferred location constraints can be reported with the **report_macro_constraints** command. Note that this command works incrementally and does not overwrite existing preferred location constraints. Use the **remove_macro_constraints** command to delete any existing preferred location constraints.

EXAMPLES

The following example captures of the macro locations in one session and applies them in another session.

```
prompt> set macros [get_cells -filter "design_type == macro" -hier *]
prompt> derive_preferred_macro_locations $macros -file preferred.tcl
```

The tool writes the preferred.tcl file, which contains **set_macro_constraints** commands for each specified macro.

```
set_macro_constraints \
  -preferred_location {0.261 0.858} \
  [get_cells I_ORCA_TOP/I_PCI_TOP/I_PCI_WRITE_FIFO/PCI_FIFO_RAM_7]

set_macro_constraints \
  -preferred_location {0.287 0.817} \
  [get_cells I_ORCA_TOP/I_PCI_TOP/I_PCI_WRITE_FIFO/PCI_FIFO_RAM_6]
...
```

The following example loads the constraints generated in the previous example. Source the preferred placement constraints before running the **create_placement -floorplan** command.

```
prompt> source preferred.tcl
prompt> report_macro_constraints $macros
****************************************
Report : report_macro_constraints
Design : ORCA
****************************************

 macro            preferred
 name             location
 --------------------------------------------------------------------------------
I_ORCA_TOP/I_PCI_TOP/I_PCI_WRITE_FIFO/PCI_FIFO_RAM_7
                  {0.261 0.858}
I_ORCA_TOP/I_PCI_TOP/I_PCI_WRITE_FIFO/PCI_FIFO_RAM_6
                  {0.287 0.817}
```

SEE ALSO

create_placement(2)
remove_macro_constraints(2)
report_macro_constraints(2)
set_macro_constraints(2)

derive_route_connection

Derive connection of a set of disjoint dangling routes.

SYNTAX

status **derive_route_connection**
list_of_nets

Data Types

list_of_nets collection

ARGUMENTS

list_of_nets

List of nets to derive.

DESCRIPTION

This command is used to derive connection of disjoint dangling routes.

It will connect these routes to make connected routes from driver to loads. It will derive terminal for un-assigned block pins at end of route inside their blocks. It will break multiple points paths into two points paths.

Deriving terminal will fail if there are multiple un-assigned block pins of same block. Please manually create terminals before using this command. It may also fail if there are multiple ends of wires in that block, because this command does not know end of which wire is the expected location of the terminal. If user has multiple wire ends inside the block, please make sure other wire ends, which are not expected to be derived as the terminal, be close enough to the wire it will connect.

When connecting routes, this command just follows shortest path without honor anything, e.g. routing rules, blockages or routing DRCs.

EXAMPLES

The following examples show usage of command options.

prompt> **derive_route_connection {net_1 net 2}**

SEE ALSO

add_buffer_on_route(2)

describe_state_transition

Specifies the legality of a transition from one object's named power state to another.

SYNTAX

status **describe_state_transition**
transition_name
-object *object_name*
[-from *from_list* -to *to_list*]
[-paired {*from_state to_state*}]
[-through *through_list*]
[-legal]
[-illegal]

Data Types

transition_name string
object_name string
from_list list
to_list list
through_list list
from_state string
to_state string

ARGUMENTS

transition_name

Specifies the simple name of transition.

-object *object_name*

Specifies simple name of any UPF object which contains the power state.

-from *from_list*

Specifies an unordered list of power state names active before a state transition. Empty list expands to all named power states for the specified object.

-to *to_list*

Specifies an unordered list of power state names active after a state transition. Empty list expands to all named power states for the specified object.

-paired {*from_state to_state*}

Specifies a list of from-state name and to-state name pairs.

-through *through_list*

Specifies list of intermediate states. This option is only applicable when -from/-to options are specified. The effective transition sequence is -from, -through, -to.

-legal | **-illegal**

Specify the legality of the transition being defined as either legal or illegal. The default is -legal.

DESCRIPTION

The describe_state_transition command specifies the legality of a transition from one object's named power state to another.

This command will be deprecated in UPF 3.0 and be replaced by add_state_transition.

EXAMPLES

The following example shows how to describe an illegal state transition, turn_on:

```
prompt> describe_state_transition turn_on \
 -object PdA -from {SLEEP_MODE} -to {HIGH_SPEED_MODE} -illegal
```

SEE ALSO

add_power_state(2)
add_state_transition(2)

dft_drc

Checks the current design against test design rules.

SYNTAX

status **dft_drc**
[-test_mode test_mode_label]
[-verbose]
[-coverage_estimate]
[-gui]

Data Types

test_mode_label string

ARGUMENTS

-test_mode

Specifies the mode in which we are running dft_drc. Examples of options are Internal_scan and ScanCompression_mode

-verbose

Controls the amount of detail when displaying violations. If specified, every violation instance is displayed. By default, only the first instance and the number of instances are displayed.

-coverage_estimate

Generates a test coverage estimate at the end of post-DFT DRC design rule checking.

-gui

Opens the DRC violation browser in the application GUI. The GUI is started if not already running.

DESCRIPTION

This command checks the current design against the test design rules used for scan insertion. If there are unconnected test modes and functional Autofix clocks, they are constrained as needed.

If design rule violations are found, the appropriate messages are generated.

Perform test design rule checking on a design before performing any other DFT operations, such as **insert_dft**.

This command requires the existence of a valid test protocol. The test protocol can be generated using the **create_test_protocol** command.

The severity of violations falls under one of the following categories:

- Information indicates no action is required.
- Warning indicates that you should analyze the violations. These might violate sequential cells resulting in their exclusion from scan chains. However, they do not prevent you from running certain DFT Compiler commands.
- Fatal violations stop you from proceeding. Certain DFT Compiler commands cannot be run.

If you start with a design with pre-existing scan structures, the scan information for inference must be specified with **-view** as **existing_dft** for the **set_dft_signal** and **set_scan_path** commands and run the **create_test_protocol** command.

The \fb-coverage option generates test coverage statistics for the current design. The option does not allow you to save or write out test patterns. The number generated is only an estimate and might be different from the one generated by an ATPG tool.

If you are running the command in the Design Vision environment, you can use the violation browser to analyze violations.

EXAMPLES

The following command performs test design-rule checking for the current design, and illustrates the types of messages that are printed:

```
dft_drc -verbose
Information: DFT DRC prelude...
Information: Initializing...
Information: Design verilog ./TOP_26795.21312.v...

Printing model sub to verilog file
Finished printing verilog ./_sub_26795.21312.v
Saved under the hood file
Information: Core vLog models...
Information: temp_model.ctl file deleted...
Information: Internal pins protocol ./TOP_26795.21312.spf for mode all_dft...
Information: TMAX script ./TOP_26795.21312.tmax...
Information: Running Pre-DFT DRC for mode all_dft
Information: Connecting socket...
Information: Lauching TMAX DRC Engine...
Information: Closing socket...
 ------------------------------------------------------------------------------
 Begin prescan design rules checking...
 ------------------------------------------------------------------------------
 Begin reading test protocol file ./TOP_26795.21312.spf...
 End parsing STIL file ./TOP_26795.21312.spf with 0 errors.
 Test protocol file reading completed, CPU time=0.00 sec.
 ------------------------------------------------------------------------------
 Begin simulating test protocol procedures...
 Test protocol simulation completed, CPU time=0.00 sec.
 ------------------------------------------------------------------------------
 Begin prescan clock rules checking...
 Prescan clock rules checking completed, CPU time=0.00 sec.
```

```
---------------------------------------------------------------------------
Begin DRC dependent learning...
Fast-sequential depth results: control=2(12), observe=0(0), detect=3(7), CPU time=0.00 sec
DRC dependent learning completed, CPU time=0.00 sec.
---------------------------------------------------------------------------
Prescan DRC Summary Report
---------------------------------------------------------------------------
No violations occurred during prescan DRC process.
Design rules checking was successful, total CPU time=0.00 sec.
```

SEE ALSO

create_test_protocol(2)
current_design(2)
gui_start(2)
insert_dft(2)
preview_dft(2)
set_dft_signal(2)

disconnect_3d_bumps

Removes logical connections between bumps and I/O drivers.

SYNTAX

status **disconnect_3d_bumps**
[-nets *nets*]

Data Types

nets list

ARGUMENTS

-nets *nets*

Specifies the nets for logical connection removal. By default, logical connections for all top-level nets between bumps and I/O drivers are removed.

DESCRIPTION

This command removes logical connection between bumps and I/O driver.

EXAMPLES

The following example removes the logical connections between bumps and I/O ports for net VDD.

prompt> **disconnect_3d_bumps -nets VDD**

SEE ALSO

create_3d_mirror_bumps(2)
propagate_3d_connections(2)
set_3d_chip_placement(2)

disconnect_net

Disconnects a net or scalar nets of net_bus from ports, scalar ports of port_bus and pins.

SYNTAX

int **disconnect_net**
[-design *design*]
[-net *net*]
[*net*]
[-all]
terms

Data Types

design collection
net list
terms list

ARGUMENTS

-design *design*

Specifies the design in which to disconnect the net or net_bus. If no design is specified, the net or net_bus is disconnected in the current design.

-net *net*

Specifies the net or net_bus to be disconnected. It can be a name or collection. The net can be a scalar (single-bit) net or net_bus and must exist in the current design. If net_bus is specified along with terms, then terms must contain the same width port_bus or length of the scalar port list must be same as bus width of the net_bus.

net

Specifies the net or net_bus to be disconnected. It can be a name or collection. The net can be a scalar (single-bit) net or net_bus and must exist in the current design. If net_bus is specified along with terms, then terms must contain the same width port_bus or length of the scalar port list must be same as bus width of the net_bus. This positional argument is provided for compatibility with other tools.

-all

Specifies to break all connections on the net.

terms

Specifies the ports,port_bus and pins to disconnect from the net. Only objects existing in the current design can be specified.

DESCRIPTION

The **disconnect_net** command breaks the connections between the specified ports,scalar ports of port_bus and pins from its net. The net, net_bus, ports,port_bus and pins must reside in the same hierarchy. The net, net_bus, pins,ports and port_bus are not removed.

Either *terms* or **-all** must be specified. If **-all** is specified, then **-net** must be specified as well.

If the *terms* is specified without the **-net** option, then the specified ports, scalar ports of port_bus and pin are disconnected from its currently connected net.

If a port or pin cannot be disconnected, the port or pin is skipped. And a warning message will be printed. The remaining ports and pins in the *terms* are disconnected.

The number of disconnected ports and pins are returned.

To connect nets, use the **connect_net** command. To display the ports and pins connected to a net, use the **all_connected** command.

EXAMPLES

The following examples show nets being disconnected using the **disconnect_net** command.

```
prompt> disconnect_net -net NET0 [get_ports A1]
prompt> disconnect_net [get_pins U1/A]
```

The following example shows all connections on a net being broken using **disconnect_net**.

```
prompt> disconnect_net -net MY_NET_1 -all
prompt> all_connected [get_nets MY_NET_1]
{}
```

The following example disconnects net_bus *N[7:0]* to port_bus *P[7:0]* so that N[7] is disconnected from P[7], N[6] is disconnected from P[6] ..,N[0] from P[0].

```
prompt> disconnect_net -net N P
prompt> 8
prompt> all_connected [get_net_buses N]
{}
prompt> all_connected [get_port_buses P]
{}
```

The following example shows the result of disconnecting an open pin *OPEN_PIN* (the pin which is not connected to any net) from net *NET*.

```
prompt> disconnect_net -net NET OPEN_PIN
Warning: Object 'OPEN_PIN' is not connected to net 'NET'. (NDMUI-916)
```

SEE ALSO

connect_net(2)
all_connected(2)

distribute_objects

Distributes the specified objects.

SYNTAX

status **distribute_objects**
[*object_list*]
[-anchor *object_list*]
[-parent]
[-to {*x1 y1*}]
[-to_box {{*x1 y1*} {*x2 y2*}}]
[-side left | right | top | bottom | hcenter | vcenter]
[-anchor_side auto | ll | ur | center]
[-spacing *spacing*]
[-pitch *pitch*]
[-group]
[-margin]
[-space_anchor]

Data Types

object_list collection
x1 float
y1 float
x2 float
y2 float
spacing float
pitch float

ARGUMENTS

object_list

Collection or selection set containing objects to distribute. If this option is not specified, global selection set is used.

-anchor *object_list*

Specifies the first object that other objects are distributed from.

-parent

Specifies that objects are distributed starting at the parent edge.

For terminals and I/O pads the parent object is the die. For soft macro pins the parent object is the soft macro. For all other objects the parent is the core.

-to {*x1 y1*}

Specifies that objects are distributed starting at the specified position.

-to_box {{*x1 y1*} {*x2 y2*}}

Specifies that objects are distributed starting at the specified rectangle.

-side left | **right** | **top** | **bottom** | **hcenter** | **vcenter**

Specifies the side of the object to be used for placing distributed objects. The valid values are as follows:

left - Use the left side
right - Use the right side
top - Use the top side
bottom - Use the bottom side
hcenter - Use the horizontal center
vcenter - Use the vertical center

The default is **left**.

-anchor_side auto | **ll** | **ur** | **center**

Specifies anchor object side to be distributed from. The valid values are as follows:

auto - Determine the anchor side automatically
ll - Use the low left side
ur - Use the upper right side
center - Use the center

The default is auto.

-spacing *spacing*

Specifies the spacing between distributed objects. The default is **0.0**.

-pitch *pitch*

Specifies the pitch between distributed objects. The default is **0.0**.

-space_anchor

Specifies that spacing should be also applied for placement of objects relatively to the anchor.

-group

Specifies that objects should be treated as if it was single object.

-margin

Use extended objects boundaries. The block boundaries are extended with keepout margins. The voltage areas boundaries are extended with guard bands. To exclude certain block keepout margins from consideration, use the following command:

```
prompt> win_set_filter -class cell \
  -filter {hard_macro_margin hard_margin route_blockage_margin soft_margin}
```

DESCRIPTION

This command distributes a list of unfixed objects to place them side by side horizontally or vertically with an optional spacing between each object.

You can specify an anchor object, anchor position, or let the command automatically determine the anchor object from the supplied list of objects using the following algorithm:

- If any objects are marked as fixed, the anchor objects is selected from the set of fixed objects. The rightmost, leftmost, bottommost or topmost fixed object for alignment of left, right, top and bottom respectively.
- If no objects are marked as fixed, the anchor object is the leftmost, rightmost, topmost or bottommost object for alignment of left, right, top and bottom respectively.

NOTES

Snapping is done automatically using global snap settings.

EXAMPLES

The following example moves the currently selected objects so that they are placed side by side in the horizontal direction with 20 units between each.

```
prompt> distribute_objects -parent -spacing 20
```

The alternative syntax uses a collection to specify the objects.

```
prompt> distribute_objects [get_selection] -parent -spacing 20
```

SEE ALSO

align_objects(2)
change_selection(2)
get_edit_setting(2)
get_selection(2)
get_snap_setting(2)
set_edit_setting(2)
set_snap_setting(2)
snap_objects(2)
spread_objects(2)

drive_of

Determines the drive resistance of the specified library cell pin.

SYNTAX

float **drive_of**
[-rise]
[-fall]
lib_cell_pin

Data Types

lib_cell_pin collection

ARGUMENTS

-rise

Gets rise drive value.

-fall

Gets fall drive value.

lib_cell_pin

Specifies the name of the library cell pin, or a collection that contains the library cell pin, for which to get the drive resistance. But the tool wrill only return the drive resistance for first lib pin in the collection.

DESCRIPTION

The **drive_of** command returns the drive resistance of of the first given library cell pin. If neither the *-rise* nor *-fall* option is specified, the greater value of rise and fall is returned. The calculation of the driver resistance is based on current corner.

echo

Echos arguments to standard output.

SYNTAX

string **echo**
  [-n]
  [*arguments*]

Data Types

arguments string

ARGUMENTS

-n

Suppresses the new line. By default, **echo** adds a new line.

arguments

Specifies the arguments to be printed.

DESCRIPTION

The **echo** command prints out the value of the given arguments. Each of the arguments are separated by a space. The line is normally terminated with a new line, but if the **-n** option is specified, multiple **echo** command outputs are printed onto the same output line.

The **echo** command is used to print out the value of variables and expressions in addition to text strings. The output from **echo** can be redirected using the **>** and **>>** operators.

EXAMPLES

The following are examples of using the **echo** command:

prompt> **echo**

```
"Running version" $sh_product_version
Running version v3.0a

prompt> echo -n "Printing to" [expr (3 - 2)] > foo
prompt> echo " line." >> foo
prompt> sh cat foo
Printing to 1 line.
```

SEE ALSO

sh(2)

eco_netlist

Performs name-based netlist comparison between a working design and the golden input.

SYNTAX

status **eco_netlist**
  -write_changes *write_changes_file_name* |
    -write_changes_per_module *directory_name*
  { [-by_verilog_file *verilog_file_name*] |
    [-by_block *golden_block[/label_name]*
    [-golden_lib *golden_library*]
    [-working_block *working_block[/label_name]*]
    [-working_lib *working_library*]
    ]
  }
  [-compare_physical_only_cells]
  [-extract_timing_eco_changes]
  [-compare_target_modules *target_module_name_list* |
    -compare_module_subsets | -top_module *module_name*]
  [-write_summary *summary_file_name*]
  [-compare_pg]
  [-cross_physical_hierarchy]

Data Types

write_changes_file_name string
directory_name string
verilog_file_name string
golden_block string
label_name string
golden_library string
working_block string
working_library string
target_module_name_list list
summary_file_name string
module_name string

ARGUMENTS

-write_changes *write_changes_file_name*

Specifies a file name to write the comparison result in. This option is mutually exclusive with **-write_changes_per_module**. You must specify one of this option or **-write_changes_per_module**.

-write_changes_per_module ***directory_name***

Writes changes for each module separately. The change files are written to the specified directory with the module name as the file name. This option is mutually exclusive with **-write_changes**. You must specify one of this option or **-write_changes**.

When this option is specified, a source file "source_sequence.tcl" will be created for each block, which can be used to source all changes per module in this block with the correct sequence. When this option is used with **-cross_physical_hierarchy**, a single file named "full_chip_source_sequence.tcl" will be created under the specified directory, which can be used to source all changes per block with the correct sequence.

-by_verilog_file ***verilog_file_name***

Specifies a Verilog file as the golden input. You must specify this option or by_block. They are mutually exclusive with each other. The Verilog file can contain part of modules when **-compare_target_modules** or **-compare_module_subsets** is specified.

-by_block ***golden_block/label_name***

Specifies an existed block as the golden input. It is optional to append a label name after the block name. You must specify this option or by_verilog_file. They are mutually exclusive with each other.

-golden_lib ***golden_library***

Specifies the library which the golden block is in. It is optional and if it is not specified, the golden block will be found in current library. This option must be used together with **-by_block**.

-working_block ***working_block/label_name***

Specifies the working block and optional with a label name. It is optional and can let you choose a non-current block as the working block. If this option is not specified, the current block will be used as working block. This option must be used together with by_block.

-working_lib ***working_library***

Specifies the library name which the working block is in. It is optional and if unspecified, the working block will be found in current library. This option must be used together with **-by_block** and **-working_block**.

-compare_physical_only_cells

Specifies that physical-only cells are included in comparison. By default, this option is off and the command will ignore the changes of physical-only cells.

-extract_timing_eco_changes

Extracts the timing ECO changes such as **size_cell**, **add_eco_repeater** and **remove_eco_repeater**. Except the timing ECO changes, if there still are some functional ECO changes between the working block and golden input, the command will also write out the changes together with the timing ECO changes.

-compare_target_modules ***target_module_name_list***

Compares only the specified modules. This option is mutually exclusive with **-compare_module_subsets** and **-top_module**.

-compare_module_subsets

Compares only the modules provided in the golden input. This option is mutually exclusive with **-compare_target_modules** and **-top_module**.

-top_module

Specify the top module for loading the golden verilog file when there are multiple top modules in this verilog file. This option is mutually exclusive with **-compare_target_modules** and **-compare_module_subsets**.

-write_summary ***summary_file_name***

Specifies the file name to use for writing a detail summary of the comparison. By default, the command displays only a simple comparison result.

-compare_pg

Specifies that PG objects (nets, ports, pins) are included in the comparison. Comparison rules for non-PG objects are used. By default, PG objects are excluded.

-cross_physical_hierarchy

Specifies that the command crosses physical hierarchy and also compares the child blocks. This option must be used together with **-write_changes_per_module** and it is mutually exclusive with **-write_changes**. By default, this option is off and the comparison will not cross physical hierarchy.

DESCRIPTION

This command identifies netlist changes in Verilog flow or tracks netlist changes before and after certain operations. This command does a name-based comparison between working and golden and only logical netlist changes are involved. The working and golden designs must be synthesized. This command will not apply the comparison result on working block. You should source the output change list manually if you want to apply the changes on working block.

This command can output the netlist changes in Tcl format of module-based ECO editing commands. You must specify the file name or the directory name for the output.

EXAMPLES

The following example shows the usage of input Verilog file as golden.

```
prompt> eco_netlist by_verilog_file after_eco.v \
  -write_changes eco.tcl
```

The following example show the usages of input existed block as golden. Use the current block as working and another block in current library as golden.

```
prompt> eco_netlist -by_block golden_block \
  -write_changes eco.tcl
```

The following example uses the current block as working and another block in another library as golden.

```
prompt> eco_netlist -by_block golden_block \
  -golden_lib golden_library -write_changes eco.tcl
```

The following example uses the block in another libraries as the working and golden.

```
prompt> eco_netlist -by_block golden_block \
  -golden_lib golden_library -working_block working_block \
  -working_lib working_library -write_changes eco.tcl
```

The following example uses the blocks with label name.

```
prompt> eco_netlist -by_block golden_block/after_buffering \
  -write_changes eco.tcl
```

The following example shows the usages of involving physical only cells in comparison.

```
prompt> eco_netlist -by_block golden_block/spare_cells \
  -compare_physical_only_cells -write_changes eco.tcl
```

The following example shows the usage of extracting timing ECO changes.

```
prompt> eco_netlist -by_block golden_block \
  -extract_timing_eco_changes -write_changes eco.tcl
```

The following examples show the usage of doing partial netlist diff to compare module A and B

```
prompt> eco_netlist -by_verilog_file golden.v \
  -write_changes eco.tcl -compare_target_modules {A B}
```

The following example compares modules in golden as subset modules:

```
prompt> eco_netlist -by_verilog_file golden.v \
  -write_changes eco.tcl -compare_module_subsets
```

The following examples show the usage of the summary options.

```
prompt> eco_netlist -by_verilog_file golden.v \
  -write_changes eco.tcl -write_summary eco.sum
```

The detail summary in file eco.sum is as follows:

```
=========================
eco_netlist Diff Summary
=========================
Total 2 modules compared, 100.00% modules have difference.
  0 modules created.

  0 modules deleted.

  2 modules changed.
    Module sub has max changes: 6
    Average changes per module: 6
    sub: 6 changes (60.00%)
      Instance changed  : 0 (0.00%)
      Port changed      : 3 (150.00%)
      Connection changed: 3 (50.00%)
    top: 6 changes (60.00%)
      Instance changed  : 0 (0.00%)
      Net changed       : 0 (0.00%)
      Port changed      : 3 (150.00%)
      Connection changed: 3 (50.00%)
```

The following examples show the usage of **-write_changes_per_module**.

```
prompt> eco_netlist -by_verilog_file golden.v -write_changes_per_module changes
```

Consider the case where there are two modules, A and B, that have differences. The preceding command generates a directory named "changes". In the directory, the files A.tcl and B.tcl contain the changes of module A and B respectively. The file source_sequence.tcl contains the source commands in correct sequence.

The file A.tcl might look like this:

```
# sourcing follow files in order before sourcing this file
# B.tcl
```

```
  [changes of module A]
```

The file source_sequence.tcl might look like this:

```
source B.tcl
source A.tcl
```

The dependency is represented by indent. In this example it means B is a sub module of A and you must source B.tcl before sourcing A.tcl.

SEE ALSO

add_eco_repeater(2)
change_link(2)
connect_net(2)
create_cell(2)
create_module(2)
create_net(2)
create_port(2)
disconnect_net(2)
edit_module(2)
remove_cells(2)
remove_eco_repeater(2)
remove_nets(2)
remove_ports(2)
size_cell(2)
source(2)

eco_opt

Optimizes the current routed design with PrimeTime timing and PrimeTime ECO.

SYNTAX

status **eco_opt**
[-types *type_list*]
[-pba_mode none | path | exhaustive]
[-physical_mode open_site | occupied_site]
[-training_data_directory *directory_name*]
[-pre_eco_script *file_name*]
[-eco_script *file_name*]
[-write_change_file_only]
[-no_eco_place_and_route]
[-hold_margin *margin*]
[-setup_margin *margin*]
[-punch_port]
[-save_session *directory_name*]
[-size_only]

Data Types

type_list list
directory_name string
file_name string
margin float

ARGUMENTS

-types *type_list*

Specifies the optimization types. Specify one or more of the following optimization types:

- **timing** - Fix setup and hold timing violations.
- **setup** - Fix setup timing violations.
- **hold** - Fix hold timing violations by selecting hold buffers and delay cells and passing them to PrimeTime ECO.
- **sequential_timing** - Fix setup and hold violations for sequential cells. Must be specified with the **setup**, **hold**, or **timing** type.
- **useful_skew** - Fix setup and hold timing violations on the clock network.
- **drc** - Fix maximum transition and maximum capacitance violations.
- **max_transition** - Fix maximum transition violations.

- **max_capacitance** - Fix maximum capacitance violations.
- **max_clock_transition** - Fix maximum transition violations on the clock network.
- **leakage_power** - Reduce leakage power by sizing and swapping cells. The tool selects the worst leakage power scenario based on PrimeTime-PX power analysis, and passes the information to PrimeTime ECO for power optimization.
- **dynamic_power** - Reduce dynamic power by downsizing cells. The tool selects the worst dynamic power scenario based on PrimeTime-PX power analysis, and passes the information to PrimeTime ECO for power optimization.
- **total_power** - Reduce total power by sizing and swapping cells. The tool selects the worst total power scenario based on PrimeTime-PX power analysis, and passes the information to PrimeTime ECO for power optimization.
- **buffer_removal** - Remove redundant buffers for power reduction.
- **noise** - Fix noise violations.
- **power_integrity** - Fix power integrity violations. User will need to provide path to rail database generated by RedHawk analysis to use this feature. Cannot be combined with other fixing types.

When multiple types are specified, the tool determines the optimum order of the different types of optimization.

The default is **{setup hold drc total_power buffer_removal}**. The default DRC fixing includes both **max_transition** and **max_capacitance**.

-pba_mode none | path | exhaustive

Controls the use of path-based analysis (PBA). The effort value can be

- **none** (the default)

  Performs graph-based analysis (GBA).

- **path**

  Path-based analysis is applied to paths after they are gathered.

- **exhaustive**

  An exhaustive path-based analysis path search algorithm is applied to determine the worst path set in the block.

These meanings are the same as in the PrimeTime **report_timing** and **get_timing_paths** commands.

Path-based analysis (PBA) removes pessimism and provides more accurate timing but spends more time than graph-based analysis (GBA). When the current block is optimized with PBA, it can reduce power more than GBA-based optimization as more positive slacks are available for downsizing or swapping. PBA-based optimization can also reduce the number of changes as the number of violations is smaller than the number of violations in GBA.

PBA-based optimization might increase runtime. Moreover, PBA runtime might increase if your block has a large number of violations.

When this option is specified, the fixing process uses the specified path-based analysis type to determine the timing fixes needed. If combined with the **-eco_script** option for user-specified PrimeTime ECO commands, the **-pba_mode** determines only the path-based analysis type for the built-in QoR reports within the **eco_opt** command, not affecting the user-specified script. In this way, the intermediate QoR reports' path-based analysis mode match the user-specified ECO script.

-physical_mode none | open_site | occupied_site

Specifies the use of physical data to resize cells or place buffers.

The valid values are

- **none**

  Does not use physical data to resize cells or place buffers.

- **open_site** (default)

  Sizes a cell if there is enough room available around the cell and inserts a buffer if there is an empty site. This mode avoids moving existing cells; it places cells only in empty sites.

- **occupied_site**

  The PrimeTime tool considers the layout density in the cell neighborhood and can place or resize a cell overlapping existing neighbor cells when the local placement density can accommodate the change. This mode places or resizes a cell even if no empty site is available; later place and route commands move existing cells to create room for the new or sized cells and change routing accordingly for the best results.

-training_data_directory *directory_name*

Specifies the directory for storing the training data for PrimeTime power optimization.

-pre_eco_script *file_name*

Specifies a Tcl script file to be sourced in the PrimeTime tool before the **check_eco** command is executed.

-eco_script *file_name*

Specifies a file that contains PrimeTime ECO commands.

The following example shows a custom ECO script file that contains PrimeTime ECO commands and a custom Tcl procedure, apply_my_custom_dont_touch.

```
# apply custom dont touch
apply_my_custom_dont_touch
# perform ECO
fix_eco_drc
fix_eco_timing -type setup -slack_greater_than 0.1
fix_eco_power -pattern {HVT LVT}
```

When this option is specified, the **-types** option is ignored. Instead, the current block is optimized with the PrimeTime commands specified in the file.

-write_change_file_only

Writes the PrimeTime ECO changes to a file instead of sourcing them.

Use this option carefully as the design status in the immplementation tool would be different from that of the PrimeTime tool.

-no_eco_place_and_route

Does not perform ECO placement and routing after making changes.

By default, the command runs ECO placement and routing.

-setup_margin *margin*

Specifies a margin applied to setup timing slack during PrimeTime ECO fixing, in library time units.

-hold_margin *margin*

Specifies a margin applied to hold timing slack during PrimeTime ECO fixing, in library time units.

-punch_port

Performs port punching for maximum transition and noise violation fixing.

-save_session ***directory_name***

Specifies the directory in which to save the PrimeTime session.

-size_only

Only sizing methods will be deployed with the specified fixing types. **max_fanout** and **buffer_removal** fixing types will not be executed when *-size_only* is specified.

DESCRIPTION

This command optimizes the current block with PrimeTime timing and PrimeTime ECO features. By default, the command invokes the PrimeTime tool, runs PrimeTime ECO including timing and DRC fixing as well as power optimization, brings back the changes to the implementation tool, followed by ECO placement and routing. The command also provides the ability to specify custom PrimeTime ECO commands for flexibility and to meet complex ECO requirements.

When the **eco_opt** command executes, the following actions happen:

- Generates PrimeTime setup based on the tool's multi scenario
- Invokes Distributed Multiscenario Analysis (DMSA) PrimeTime under the hood unless PrimeTime has already started
- Runs multiscenario timing updates
- Performs the specified types of PrimeTime ECO
- Brings the changes back to the implementation tool by sourcing the change list generated by the PrimeTime tool
- Performs incremental placement and routing with the recommended minimal physical impact (MPI) settings

EXAMPLES

The following example performs setup fixing followed by incremental place and route.

```
prompt> eco_opt -types setup
```

The following example performs setup and hold timing fixing followed by incremental place and route, and checks the PrimeTime timing with the **check_pt_qor** command.

```
prompt> eco_opt -types {setup hold}
prompt> check_pt_qor
```

The following example performs total-power optimization with path-mode PBA.

```
prompt> eco_opt -types total_power -pba_mode path
```

The following example performs the custom ECO commands defined in the my_eco.tcl file and checks the PrimeTime QoR.

```
prompt> eco_opt -eco_script my_eco.tcl
prompt> check_pt_qor -type summary
```

The following example runs the predetermined fixing types followed by a custom ECO script. Note that placement and routing are

performed only once, after the last command that sources all changes made by the three **eco_opt** commands.

```
prompt> eco_opt -types setup -write_change_file_only
prompt> eco_opt -types total_power -write_change_file_only
prompt> eco_opt -eco_script my_pteco.tcl
```

SEE ALSO

check_pt_qor(2)
legalize_placement(2)
report_pt_options(2)
report_starrc_options(2)
route_eco(2)
set_pt_options(2)
set_starrc_options(2)

eco_update_supply_net

This command updates supply nets for eco buffers.

SYNTAX

status **eco_update_supply_net**
[-cells *cells*]

Data Types

cells collection

ARGUMENTS

-cells *cells*

Specify cells to update supply nets. The specified cells and the cells in their "eco buffer trees" will be updated. Here "eco buffer tree" means a group of connected timing eco cells, separated by cells of other types. If this option is not specified, all cells added by timing eco commands will be updated, including add_buffer, add_eco_repeater, split_fanout, add_buffer_on_route and size_cell(in freeze_silicon).

DESCRIPTION

This command update supply nets for eco buffers. If there is any violation when updating the MV supply nets, the command will give warning messages and not apply supply nets for related cells. But the command will not remove the cells which have violations during updating supply nets.

The command may handle multiple buffer trees, if some trees failed, the failed cells will be added to the collection $ecoUpdateSupplyFailedCells, and for each failed buffer tree, a warning message will be issued .

This command does not support undo.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples shows how to using the eco_update_supply_net command to update supply nets for the specified cells.

prompt> **eco_update_supply_net -cells {eco_cell_0}**

The following examples shows how to using the eco_update_supply_net command to update supply nets for all eco buffers which have no supply updated yet.

prompt> **eco_update_supply_net**

SEE ALSO

add_buffer_on_route(2)
check_mv_design(2)

edit_block

Run specified commands editing the block

SYNTAX

edit_block
command_list

Data Types

command_list string list

ARGUMENTS

command_list

Specifies the design modification commands to run.

DESCRIPTION

The **edit_block** command allows a list of commands to run within one together, minimizing the runtime needed to for computation to keep the database consistent, which is normally done at the end of every command.

Certain operations, for example, collection cleanup are required to be performed at the end of every command, and can get expensive when a lot of collections are active. Using **edit_block** will help in such cases as well.

Following are the eco commands supported by **edit_block**:

- create_cell
- remove_cell
- change_link
- create_port
- remove_port
- create_net

- remove_net
- connect_net
- disconnect_net
- set_attribute
- remove_port_buses
- create_port_bus
- remove_net_buses
- create_net_bus
- add_eco_repeater
- remove_eco_repeater
- size_cell

The usage of command **set_attribute** in **edit_block** is limited to netlist objects only.

EXAMPLES

The following example edits the block, creating a new cell and connecting a net to one of the pins of the new cell.

```
prompt> edit_block {
  create_cell iAnd [get_lib_cells lib/AND2]
  connect_net -net n2 iAnd/a1
}
```

The following example runs the source command

```
prompt> edit_block {
  source -e -v  commands.tcl
}
```

SEE ALSO

edit_module(2)

edit_ems_rule

Edits an user-defined EMS rule.

SYNTAX

edit_ems_rule
-name *rule_name*
[-severity *rule_severity*]
[-message *message_template*]
[-parameters *parameter_properties*]

Data Types

rule_name string
rule_severity string
message_template string
parameter_properties string

ARGUMENTS

-name *rule_name*

This is a mandatory option. This name is used to identify the rule to be edited. The rule name should obey regular expression [A-Z]+[-][0-9]+ i.e one or more upper-case alphabets, followed by hyphen character and which should be followed by one or more numerals. If rule name does not obey this regular expression then an error message is displayed. User should have already created an EMS rule by this name using command **create_ems_rule** before using this command. If an EMS message is already created using this rule then the rule is considered frozen. One cannot edit a frozen rule, it results in an error message.

-severity *rule_severity*

This is an optional option. This option is used to specify whether the rule corresponds to an error, warning or information. The valid string arguments are **error**, **information**, **info**, **warning** and **warn**. These string values can be in upper or lowercase or a mix of both. But it is advisable to stick to one convention for the sake of uniformity. But when the EMS message is displayed in the shell or in the EMS GUI message browser, the severity string displayed will be one of **Error**, **Warning** or **Information**.

-message *message_template*

This is an optional option. This option is used to specify the message template for the rule. The message template can contain placeholders referred to as **parameters**. Parameters begin with a **%** character. The parameter name should obey regular expression [A-Za-z0-9_]+ i.e it should contain one or more of upper or lower case alphabets or numerals or underscore character. There should not be any whitespace between **%** and parameter name. To use a literal percent sign (**%**), specify **%%**.

-parameters *parameter_properties*

This is an optional option. If there are no EMS parameters for the rule being defined then this option need not be used. But if there

are EMS parameters specified in message template this option can be used to define the parameter properties. This option takes a string as an argument which is a sequence of fields separated by semi-colon (;) describes properties of an EMS parameter, which again is a sequence of pairs of format <name>:<value>. Currently, an EMS parameter can have at-most three properties namely, name-property, type-property and command-property. Here type-property and command-property are optional and name-property is mandatory. name-property will take a valid parameter name as value and which helps EMS to know properties of which parameter are defined. type-property can take one of **string**, **int**, **float** and **double** values. type-property is used to define the type of the EMS parameter. The command-property of the EMS parameter can take value as any of the **get_*** commands such as get_pins, get_cells etc., which take an object name as argument and return the object handle. This property is used to get the object handle corresponding to the EMS parameter name.

DESCRIPTION

Command **edit_ems_rule** edits an already existing user-defined EMS rule. An EMS rule is a message template with some auxiliary information used to create the EMS message. The user-defined rules created are not persistent across sessions. This command can be used to add or override information related to an already created user-defined rule. But if an EMS message is already created for the user-defined rule to be edited, then editing fails displaying an error message.

EXAMPLES

```
prompt> create_ems_rule -name "TEMP-001" -severity "Info" -message "Pin:%pin has capacitance:%cap"
prompt> edit_ems_rule -name "TEMP-001" -parameters "name:pin type:string command:get_pins"
prompt> edit_ems_rule -name "TEMP-001" -parameters "name:cap type:double"
prompt> create_ems_rule -name "TEMP-002" -severity "Error" -message "Cell %cell is not placed"
prompt> edit_ems_rule -name "TEMP-002" -severity "Info"
prompt> edit_ems_rule -name "TEMP-002" -message "The Cell %cell is not placed."
prompt> edit_ems_rule -name "TEMP-002" -parameters "name:cell type:string command:get_cells"
prompt> create_ems_database a.ems
{a.ems}
prompt> set cap 0.0004
prompt> create_ems_message -rule "TEMP-001" -parameters "pin:U1/I cap:$cap"
prompt> set cell U23
prompt> create_ems_message -rule "TEMP-002" -parameters "cell:$cell"
prompt> report_ems_database
 ...
Information: Pin:U1/I has capacitance 0.0004 (TEMP-001)
Information: The Cell U23 is not placed. (TEMP-002)
```

SEE ALSO

create_ems_rule(2)
create_ems_database(2)
report_ems_database(2)
create_ems_message(2)

edit_module

Edits a Verilog module in the module aspect to modify the netlist aspects.

SYNTAX

edit_module
module_list
command_list

Data Types

module_list collection
command_list list

ARGUMENTS

module_list

Specifies the list of modules to edit for different netlist modifications. *module_list* can be specified as a string (name of objects) or the collection of objects.

command_list

Specifies the netlist modification commands to apply to the specified modules. Commands are listed sequentially with each command on one line. The commands that operate at the module scope are **create_cell**, **remove_cells**, **create_net**, **remove_nets**, **create_port**, **remove_ports**, **connect_net**, **disconnect_net**, **change_link**, **get_cells**, **get_nets** and **get_ports**.

DESCRIPTION

The **edit_module** command edits a Verilog module directly in the module scope and propagates the changes to all its instances. All the Tcl commands that are used to modify a netlist can be used within this *edit_module* scope, including creating and deleting of cells, ports and nets, connecting and disconnecting of nets and changing reference of module instance.

When cells are created or reference of module instance is changed, the corresponding hier aspect instances are set respectively with an ECO status of **create_cell** or **change_link**.

The getter commands related to cells/nets/ports (get_cells/get_nets/get_ports) can also be used within *edit_module* scope. They will behave exactly similar to hier aspect get_cells/get_nets/get_ports commands. Attributes can be queried on returned objects of these getter commands. Outputs of these getter commands can be passed on to other commands as well.

The commands passed into *edit_module* are not screened; take care with the commands that you use.

EXAMPLES

The following example edits a module 'abc' where module instance iAnd is created and net n2 is connected with pin a1 of iAnd.

```
prompt> edit_module abc {
  create_cell iAnd [get_lib_cells lib/AND2]
  connect_net -net n2 iAnd/a1
}
```

The following example edits a module 'abc' where module port A1 is created, net n2 is created and module instance iBuf's reference is changed.

```
prompt> edit_module [get_modules abc] {
  create_port A1
  create_net n2
  change_link iBuf BUF2
}
```

The following example shows usage of getter commands within **edit_module**.

```
prompt> edit_module [get_modules abc] {
  get_nets -of_objects [get_cells]
  remove_cells [get_cells iAnd]
  get_attribute [get_ports in2] port_type
}
```

SEE ALSO

create_module(2)
remove_modules(2)

edit_via_matrix

Edit a via matrix in the current design.

SYNTAX

int **edit_via_matrix**
[-delete *box*]
[-add *box*]
[-update *box*]
[-lower_mask_constraint *lower_mask*]
[-cut_mask_constraint *cut_mask*]
[-upper_mask_constraint *upper_mask*]
[-force]
[-verbose]
via_matrix

Data Types

box rectangle
via_matrix collection

ARGUMENTS

-add *bounding_box*

Specifies the box region which overlaps or touch base vias to be added. *-add*, *-delete* and *-update* are mutually exclusive options.

-delete *bounding_box*

Specifies the box region which overlaps or touch base vias to be deleted. *-add*, *-delete* and *-update* are mutually exclusive options.

-update *bounding_box*

Specifies the box region which overlaps or touch base vias whose mask(s) to be updated. *-add*, *-delete* and *-update* are mutually exclusive options.

-lower_mask_constraint *lower_mask*

Updates the lower mask constraint of the base vias specified by *-update*. Valid values are no_mask, mask_one, mask_two, mask_three, mask_four, and same_mask. No matter what the via_matrix mask_pattern is, the specified base via lower mask is to be updated as *lower_mask*.

-cut_mask_constraint *cut_mask*

Updates the cut mask constraint of the base vias specified by *-update*. Valid values are no_mask, mask_one, mask_two, mask_three, mask_four, mask_five, mask_six, mask_seven, mask_eight, mask_nine, mask_ten, mask_eleven, mask_twelve, mask_thirteen, mask_fourteen, mask_fifteen and same_mask. No matter what the via_matrix mask_pattern is, the specified base via cut mask is to be updated as *cut_mask*.

-upper_mask_constraint *upper_mask*

Updates the upper mask constraint of the base vias specified by *-update*. Valid values are no_mask, mask_one, mask_two, mask_three, mask_four, and same_mask. No matter what the via_matrix mask_pattern is, the specified base via upper mask is to be updated as *upper_mask*.

-force

Edit locked or fixed a via_matrix. By default, such via_matrixes are not re sized according to global edit settings. Edit settings can be accessed using **get_edit_setting**.

-verbose

Display verbose deletion information.

via_matrix

Specifies a via matrix to edit. It may be via matrix collections, specified using the **get_via_matrixes** command.

DESCRIPTION

This command edits a via matrix.

EXAMPLES

For example to exclude all base vias overlapping or touching the specif ed bounding-box.

```
prompt> edit_via_matrix -delete {{8.0600 9.2800} {8.2545 9.4345}} $viaMatrix
```

For example to include all base vias overlapping or touching the specif ed bounding-box.

```
prompt>edit_via_matrix -add {{8.0300 9.1850} {8.0302  9.19}} $viaMatrix
```

For example to update masks of all base vias overlapping or touching the specif ed bounding-box.

```
prompt>edit_via_matrix -update {{8.28 9.50} {8.40 9.5}} $viaMatrix -lower_mask_constraint mask_three
```

SEE ALSO

get_via_matrixes(2)
remove_via_matrixes(2)

elaborate

Builds a new design using an analyzed module, entity or configuration template from an *hdl_library*.

SYNTAX

status **elaborate**
template_name
[-architecture *arch_name*]
[-hdl_library *hdl_library_name*]
[-parameters *param_list*]

Data Types

template_name string
arch_name string
hdl_library_name string
param_list string (comma-separated list)

ARGUMENTS

template_name

Specifies the name of an analyzed module, entity, or configuration for which a template is available from the specified *hdl_library*.

-architecture *arch_name*

Specifies the name of an architecture for the entity *template_name* as declared in an analyzed VHDL source. The default architecture is the one most recently analyzed.

-hdl_library *hdl_library_name*

Specifies the name of the *hdl_library* in which to find the template. When the -**hdl_library** option is not used, or it specifies "default", then the *hdl_library_name* is taken from **hdlin.hdl_library.default_name**.

-parameters *param_list*

Specifies a string of design parameter mappings. Parameters within the list must be separated by commas. A specification can be based on parameter declaration order (for example, "8,7,5") or on parameter names (for example, "N=>8,M=>6", or "N=8,M=6"). It is acceptable to mix ordered and named parameter specifications, if the ordered parameters are listed first. The full *param_list* syntax is summarized in the "DESCRIPTION" section below.

DESCRIPTION

The **elaborate** command builds the indicated design from its analyzed (template) representation using the specified parameter value substitutions. The elaborated *block_name* becomes the **current_block** in your session; its containing library becomes the **current_lib** of the session.

The syntax of parameter specification includes VHDL conventions plus the 'h, 'b, 'd, 'o constant literal forms from Verilog. Verilog accepts strings, integers, and "verilog number" values. VHDL also allows the concatenation operator ampersand (&) between two arrays or strings. Concatenating an array with a scalar is not supported. Define each named-parameter only one time; an error message issues if a name is defined more than one time.

Parameters are not case-sensitive, so NUM is the same as num. Characters inside string-values are case-sensitive, however.

The **-parameters** option must arrive as a single Tcl string token, but there are many ways to delimit it, split it across multiple input lines, and even to let the shell compute it. Typical parameter specifications are enclosed in double quotation marks ("), but this is not advised if the specification itself includes string values. Instead it can be written as a Tcl list \{inside curly braces\}. Even when using Tcl list notation, parameter specifications must be separated by a comma (,).

The formal syntax for the -parameter option is as follows:

```
param_option::=  [ param_list ] [ suffix ]

param_list::=  param_spec [ , param_spec ]*
        |   value_spec [ , param_list ]

param_spec::=  parameter_name => value_spec
        |   parameter_name = value_spec

value_spec::=  " string_value "
        |   ' character_value '
        |   integer_value
        |   enum_value
        |   ( value_spec [ , value_spec ]* )
        |   array_value_spec & array_value_spec
        |   integer_value 'h hex_value
suffix::=      | interface port, config, & defparam info
               (not user-settable)
```

EXAMPLES

The following example builds the design *mult* with *N* parameter set to 8, and *M* set to 3 leaving the result and also the templates with which to generate other variations inside a library called ELEMENTARY.

```
prompt> create_lib /remote/projects/function_libs/ELEMENTARY
prompt> analyze -format verilog \
   ~/my_function_designs/asymmetric/multiplier.v
prompt> elaborate -hdl_library LOGICAL mult -param "N=8,M=3"
```

SEE ALSO

analyze(2)
define_design_lib(2)
set_top_module(2)

hdlin.hdl_library.default_name(3)

enable_runtime_improvements

Enables off-by-default runtime improvement features to reduce place and route flow runtimes.

SYNTAX

status **enable_runtime_improvements**

DESCRIPTION

The **enable_runtime_improvements** command enables off-by-default runtime improvement features. Some of the runtime improvements are on-by-default and do not require any special settings to enable them. However, some of the runtime improvements are not yet on-by-default and this command enables those improvements. The improvements enabled by this command will be on-by-default in a future release.

This command can be run any time in the flow including before open_block.

EXAMPLES

The following example shows usage.

```
prompt> enable_runtime_improvements
```

SEE ALSO

error_info

Prints extended information on errors from the last command.

SYNTAX

string **error_info**

ARGUMENTS

This **error_info** command has no arguments.

DESCRIPTION

The **error_info** command is used to display information after an error has occurred. Tcl collects information showing the call stack of commands and procedures. When an error occurs, the **error_info** command can help you to focus on the exact line in a block that caused the error.

EXAMPLES

This example shows how **error_info** can be used to trace an error. The error is that the iterator variable "s" is not dereferenced in the 'if' statement. It should be '$s == "a" '.

```
prompt> foreach s $my_list {
?       if { s == "a" } {
?         echo "Found 'a'!"
?       }
?     }
Error: syntax error in expression " s == "a" "
      Use error_info for more info. (CMD-013)
shell> error_info
Extended error info:
syntax error in expression " s == "a" "
   while executing
"if { s == a } {
    echo "Found 'a'"
}"
   ("foreach" body line 2)
```

```
    invoked from within
"foreach s [list a b c] {
  if { s == a } {
    echo "Found 'a'"
}
}"
 -- End Extended Error Info
```

estimate_delay

Predicts endpoint slack per scenario at post-route in pre-route stages using machine learning (ML), and drives an incremental clock-opt optimization using ML-adjusted endpoint slack.

SYNTAX

status **estimate_delay**
[-training_labels "lStage"]
[-training_features "fStage lStage"]
[-train_model "fStage lStage"]
[-enable_model "fStage lStage"]
[-disable model]
[-output_dir "dir_path"]

Data Types

lStage string
fStage string
dir_path string

ARGUMENTS

-training_labels \flStage

Collects training labels (or, ground truth data for training) and stores it based on "lStage" tag provided by the user.
 User must specify "-output_dir <dir_path>" as well.

-training_features \ffStage lStage

Collects training features from "fStage", based on endpoints collected from "lStage" tag provided by the user in "-training_labels". User must specify "-output_dir <dir_path>" as well.

-train_model \ffStage lStage

Trains model based on features collected at "fStage", and labels at "lStage", tags provided by the user in "-training_labels" and "-training_features", respectively. User must specify "-output_dir <dir_path>" as well.

-enable_model \ffStage lStage

Enables model based on "fStage", and labels at "lStage", tags provided by the user in "-training_labels" and "-training_features", respectively. User must specify "-output_dir <dir_path>" as well. First, features are extracted. Second, endpoint slack values are predicted. Third, endpoint slack values are adjusted based on model predictions. User will see QoR summary before and after ML adjustments. Last, clock_opt engine is configured to run ML incremental clock_opt. Now, the user can run "clock_opt -from final_opto -to final_opto".

-disable model

Disables ML adjustments. User will see QoR summary before and after ML adjustments are removed.

DESCRIPTION

The **estimate_delay** command applies machine learning-based (ML-based) technology to adjust endpoint slack values at pre-route to post-route / route_opt values. The command is MCMM-aware, i.e., slack values are predicted per-scenario for each endpoint.

EXAMPLES

The following command collects labels or ground truth.

prompt> **estimate_delay -training_labels "lStage" -output_dir "dir_path"**

Currently, training labels are stored in "$dir_path/mlDelay/train/tLabels.lStage.NA.{min,max}.csv".

The following command collects training features.

prompt> **estimate_delay -training_features "fStage lStage" -output_dir "dir_path"**

Currently, training features are stored in "$dir_path/mlDelay/train/tFeatures.fStage_lStage.NA.{min,max}.csv".

The following command trains model.

prompt> **estimate_delay -train_model "fStage lStage" -output_dir "dir_path"**

Currently, model is stored in
"$dir_path/mlDelay/model/mlDelay.fStage_lStage.NA.{min,max}.pkl".

The following command enables the model.

prompt> **estimate_delay -enable_model "fStage lStage" -output_dir "dir_path"**

Extracted features are stored in "$dir_path/mlDelay/predict/pFeatures.fStage_lStage.NA.{min,max}.csv".
Predicted slack values are stored in "$dir_path/mlDelay/predict/pred.fStage_lStage.NA.{min,max}.csv".

The following command disables the model.

prompt> **estimate_delay -disable_model -output_dir "dir_path"**

estimate_timing

Performs virtual in-place optimization on the current design.

SYNTAX

status **estimate_timing**
[-host_options *host_options_name*]
[-pins *list_of_pins_or_ports*]
[-nets *list_of_nets*]

Data Types

list_of_blocks list
host_options_name string

ARGUMENTS

-host_options *host_option_name*

Specifies that the specified host option should be used for distributed processing. The host option is created with the **set_host_options** command. Note that only the part of **estimate_timing** that updates child blocks' views supports distributed processing so this option's effect depends on the design condition.

-pins *list_of_pins_or_ports*

Specifies the pins or ports for corresponding paths to run timing estimation. The switch is only applicable in the incremental mode, by default estimate_timing does timing estimation on the whole design.

-nets *list_of_nets*

Specifies the nets to run timing estimation. The switch is only applicable in the incremental mode, by default estimate_timing does timing estimation on the whole design.

DESCRIPTION

Performs virtual in-place optimization on the current design by simulating the effects of removing repeaters, building repeater trees and sizing cells. The simulated effects are stored as back-annotated transition times, delays and total net capacitances on the internally created artificial corner **estimated_corner**.

User can also specify **-nets** option or **-pins** option to do incremental timing estimation only on specific nets, or on paths through specific pins or ports. There should be a previous full *estimate_timing* run on the design prior to running in the incremental mode.

When the command is issued on a hierarchical design with child blocks, it first checks whether the reference view of the child blocks are acceptable to perform the virtual optimization at the current block level. Acceptable child block views are design view and timing abstract view with child level *estimate_timing* data. The timing abstract view with child level *estimate_timing* data are generated by *create_abstract -estimate_timing* command done at the child block level.

If the tool finds any child block that does not reference to one of the acceptable views, the command would open the child block's design view and run *add_feedthrough_buffer* to buffer any unbuffered feedthrough nets, then run *create_abstract -estimate_timing* to generate the appropriate timing abstract view, use change the block instance to reference to this timing abstract view, and continue with virtual optimization. The reference view change would happen to any child block unless the child block has *set_editability -value false* setting.

The *add_feedthrough_buffer* operation, which would be performed on a child block if *estimate_timing* needs to generate a new timing abstract view for it, would change the netlist. There are several app options which can be used to control the *add_feedthrough_buffer* done inside *estimate_timing*. You can do *report_app_options plan.estimate_timing.abstract** to find them.

This command returns 1 on success, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example performs virtual in-place optimization by using the artificial corner **estimated_corner**.

```
prompt> estimate_timing
```

SEE ALSO

current_corner(2)
set_host_options(2)
create_abstract(2)
plan.estimate_timing.abstract_add_feedthrough_buffer(3)
plan.estimate_timing.abstract_feedthrough_buffer_side(3)
plan.estimate_timing.abstract_feedthrough_logical(3)
plan.estimate_timing.abstract_feedthrough_user_buffer(3)

estimate_topology_timing

Estimate the timing of topology plans using net estimation rule information.

SYNTAX

longest_delay **estimate_topology_timing**
[-detail_level *level_value*]
[-routing *type*]
[-cell *type*]
[-significant_digits *digits_range*]
[-width *width_range*]
[-full_names]
[-list]

[*topology plans*]

Data Types

longest_delay float (in time units)
level_value 0 (default), 1 or 2
type "real" or "virtual"
digits_range integer, range 1 to 8
width_range integer >= 1

topology plans collection or list of names of topology plans to estimate

estimate_topology_timing.

ARGUMENTS

-detail_level *level_value*

Determines the amount of detail printed by **estimate_topology_timing**. The default value of 0 returns the longest delay of any given stage as the return value of the **estimate_topology_timing** command with no additional output.

Detail level 1 displays a header with the name of the plan, how many bits were timed (see *-cell* and *-routing* options) and the number of endpoints (for branching paths). A separate start-to-endpoint report is displayed for each unique start-to-endpoint path (i.e. each branch of the topology plan). For each segment, the "start" and "end" points of that segment are displayed, along with the incremental timing delay computed for this segment of the plan and the total cumulative delay to this point of the stage.

A stage is a set of segments that starts at the start pin/node and ends at the end pin/node or an intermediate topology repeater of type sequential (register or latch). Buffer/inverter segments are printed separately, but they do not end a stage. Cumulative timing is reset to 0.0 at the end of each stage.

The letter "F" is appended to any segment that ends in a sequential element (topology repeater of type "sequential"). The letter "W" is appended to the stage that has the worst overall delay of any stage in the plan. Incremental delay for each branch-path is the sum of the worst delay of each separate segment of the branch-path, and is reset at each sequential element (stage). This means that the estimated stage delay may be somewhat pessimistic, if the worst bit of each segment or stage is a different bit.

Detail level 2 adds display of the worst/best/average delay found for each of the bits tested for this segment, the "virtual" wire length for this segment, total "virtual" wire length for this stage as well as each virtual wire for this segment.

If more than one plan is estimated and detail is set to > 0, the worst plan name and its delay are also reported at the end of the report.

-routing *type*

Specifies the type of routing analysis to use - real or virtual. Virtual routing simply takes the topology edge points as actual routing data for each path. In virtual mode, every topology repeater has the same routing. In real mode, the actual routing shape associated with the nets of the topology plan are used to determine horizontal and vertical wire lengths, which have their delays measured using the net estimation rule for the associated topology edge.

If this parameter is not specified, the command tries to determine if real routing analysis is possible, and if so, uses real routing analysis. If it is not available, virtual routing analysis is performed. If real routing is specified and is not available, an error will be printed. If virtual routing analysis is specified, virtual routing analysis is performed with no check for real routing.

-cell *type*

Similar to the **-routing** option, this determines if actual repeater locations or topology repeater locations are used. There are several possible operating modes, between routing and cell options. for a more detailed explanation, read the "Analysis Summary" section. If real cells exist in the netlist and are associated with the appropiate topology repeater in the topology plan (for EVERY topology repeater in the plan), and each such cell is placed, real cell analysis is possible, otherwise only virtual cell analysis is possible.

If this parameter is not specified, the command tries to determine if real cell analysis is possible, and if so, uses real cell analysis. If it is not available, virtual cell analysis is performed. If real cell is specified and is not available, an error will be printed. If virtual cell analysis is specified, virtual cell analysis is performed with no check for real cell status.

-significant_digits *digits_range*

Rounds off all delay values to the specified number of significant digits. Default is 5, valid range is from 1 to 8.

-width *width_range*

Uses this many characters to print the names of the netlist elements at each segment (pins/ports/nodes/repeaters). Default is to use only as many as necessary. This can be used to increase that size for visual appeal or standardize on a specific size. If the given width is smaller than the required size, a warning is printed and the width value is ignored.

-full_names

By default, topology node and repeater names are printed only with the name of that topology plan element. However, if a reference node is encountered (a reference to a node that is part of another plan), all elements from that other plan are merged into the current plan. All such merged elements are printed with the "full" name format: block/plan/name. Specifying the -full_names option will use the full block/plan/name format for EVERY element in the report.

-list

Instead of returning the single worst delay of the plans that were estimated, returns a Tcl list of the form "{plan1 0.15} {plan2 -1.0} ... {planN 0.325}". If an error occurs (any delay is set to -1.0), the list is still returned, but an error condition is also set. To capture the return value in this case, it is suggested to use: set rc [catch {estimate_topology_timing ...} output] and the value of the variable output will contain the list of delays.

topology plans

The default is to report on all topology plans in the current block. To estimate specific plans, including plans in other blocks, use

this option with either exact plan names or wildcards

DESCRIPTION

This command builds a simplified timing graph of a topology plan, and uses net estimation rules to estimate the delay through the various parts of this topology plan.

Each sequential repeater (register/flip flop or latch) ends a timing stage. At the end, the worst timing stage is reported as the worst delay of the topology plan.

Net estimation rule timing (similar to estimate timing and the timing ruler) is used. A net estimation rule must have valid layers set, register/buffer cells set (not "--ideal--"), a valid corner set and parasitics must be loaded in order to use this command.

Resistive and capacative values are obtained from the net estimation rule layer information, as well as the specific buffer pin and register pin delay values.

Launch and capture budgets specified on the initial topology plan are added to the first and last stage, respectively.

Empty nodes / virtual nodes / tap nodes (nodes without any pins/cells) are modeled simply as additional wires that are then associated with each of their child edges. Since these nodes are used for routing purposes, their only contribution to the computed delay is a wire-type delay. Therefore, not all nodes that are part of the original topology plan will be explicitly printed in a detail report.

Analysis Summary

There are several routing/cell modes possible for analyzing a plan.

Virtual routing / virtual cells ***In this mode, the points of the edge are used as the route points, and the origin of each node and repeaters are used as the origin. No special offsets of any kind are used, and only 1 bit per segment is timed.***

Virtual routing / real cells ***In this mode, an offset from the origin of each std cell is added as an additional horizontal and/or vertical wire to the actual node/repeater origins. In this way, std cells in a repeater array that are further from the origin of the virtual repeater will have longer delays added to estimate a worse overall timing delay. One bit per repeater is timed per segment, so the command will time all n bits of the plan.***

Real routing / virtual cells ***Since there is no actual indication of which part of the routing is associated with each repeater, a ratio of "virtual routing length" (topology edge length) to actual routing shape length is computed, and then each repeater is given the portion of the real route shapes/lengths that are associated with that repeater. For example, suppose a topology edge has a virtual length of 100, with virtual repeaters at offset 20, 50 and 75 and the routing shapes for the actual net have real length 150. Then, the first (150 / 100) * 20 = 30 units of the real route are allocated to the first repeater, the next 45 units to the second repeater, the next 37.5 units to the third repeater, and the last 37.5 units from the last repeater to the end pin/port.***

If the edge has no repeaters, then the route shapes are used, and an offset is added if the route shapes do not start/end at the pins associated with the start/end nodes of the edge.

One bit per route is timed, so the command will time all n bits of the plan.

Real routing / real cells ***Actual route shapes for all the nets of each stage are added up. All n bits of the plan are timed.***

KNOWN LIMITATIONS

Currently, all folded edges in the topology plan must have the same number of folds.

This report is NOT intended to correlate to the report_timing command, since significantly less information is used to compute the delay in order to improve the performance of the computation. The values returned by this command are only an estimate. If the

uesr inputs inappropriate values for register/buffer cells or spacing values or uses incorrect parasitic data, the computed delay values can be significantly different from actual delays.

EXAMPLES

The following example estimates topology timing for plan_1 with default detail level (level 0):

```
prompt> estimate_topology_timing topo_1
0.10882
prompt>
```

The following example estimates topology timing for plan_1,
with detail level 1, width 25 and significant digts 3. This plan has only
virtual repeaters and virtual routing, so only 1 bit is timed.

```
prompt> estimate_topology_timing topo_1 -detail 1 -width 25 -significant_digits 3
Signal Velocity Report for Topology Plan topo_1 (1 bits timed)
node3                -> TOPOLOGY_REPEATER3       Del: 0.025 Tot: 0.025 F
TOPOLOGY_REPEATER3       -> TOPOLOGY_REPEATER13      Del: 0.082 Tot: 0.082
TOPOLOGY_REPEATER13      -> TOPOLOGY_REPEATER4       Del: 0.026 Tot: 0.108 F
TOPOLOGY_REPEATER4       -> TOPOLOGY_REPEATER14      Del: 0.082 Tot: 0.082
TOPOLOGY_REPEATER14      -> TOPOLOGY_REPEATER5       Del: 0.026 Tot: 0.108 F
TOPOLOGY_REPEATER5       -> TOPOLOGY_REPEATER15      Del: 0.082 Tot: 0.082
TOPOLOGY_REPEATER15      -> TOPOLOGY_REPEATER6       Del: 0.027 Tot: 0.109 F
TOPOLOGY_REPEATER6       -> TOPOLOGY_REPEATER16      Del: 0.082 Tot: 0.082
TOPOLOGY_REPEATER16      -> TOPOLOGY_REPEATER7       Del: 0.027 Tot: 0.109 FW
TOPOLOGY_REPEATER7       -> node4                Del: 0.065 Tot: 0.065
Worst stage delay: 0.109
0.109
```

The following example estimates topology timing for plan_1, with detail
level 2 and full names. The plan has real repeaters and routing, so 64 bits
are timed.

```
prompt> estimate_topology_timing topo_1 -detail 2 -full_names
Signal Velocity Report for Topology Plan topo_1 (64 bits timed)
F/topo_1/node3           -> F/topo_1/TOPOLOGY_REPEATER3 Del: 0.12891 Tot: 0.12891 F
  (wst: 0.12891 best: 0.04398 avg: 0.07440) Len: 104.87630 Tot: 104.87620
  Wire (262.98050, 323.81000) to (262.98050, 218.93370) (L 104.87630)
F/topo_1/TOPOLOGY_REPEATER3 -> F/topo_1/TOPOLOGY_REPEATER5 Del: 0.10734 Tot: 0.10734 F
  (wst: 0.10734 best: 0.10705 avg: 0.10710) Len: 104.87620 Tot: 104.87620
  Wire (262.98050, 218.93370) to (262.98050, 114.05750) (L 104.87620)
F/topo_1/TOPOLOGY_REPEATER5 -> F/topo_1/TOPOLOGY_REPEATER6 Del: 0.10950 Tot: 0.10950 F
  (wst: 0.10950 best: 0.10813 avg: 0.10842) Len: 104.87630 Tot: 104.87630
  Wire (262.98050, 114.05750) to (262.98050, 79.50170) (L 34.55580)
  Wire (262.98050, 79.50170) to (192.66000, 79.50170) (L 70.32050)
F/topo_1/TOPOLOGY_REPEATER6 -> F/topo_1/node4           Del: 0.18843 Tot: 0.18843 W
  (wst: 0.18843 best: 0.08168 avg: 0.12996) Len: 139.59420 Tot: 139.59420
  Wire (192.66000, 79.50170) to (73.01750, 79.50170) (L 119.64250)
  Wire (73.01750, 79.50170) to (73.01750, 59.55000) (L 19.95170)
Total length: 454.22300, delay: 0.53418
Worst stage delay: 0.18843
0.18843
```

SEE ALSO

create_topology_plan(2)
get_topology_plans(2)
report_net_estimation_rules(2)
report_topology_plans(2)
set_net_estimation_rule(2)

eval_checkpoint

Creates a checkpoint on a block of Tcl commands.

SYNTAX

integer **eval_checkpoint**
checkpoint_name
tcl_commands

Data Types

checkpoint_name string
tcl_commands list

ARGUMENTS

checkpoint_name

Specifies the name of the checkpoint for the checkpoint system.

tcl_commands

Specifies the Tcl command block that is defined as a checkpoint.

DESCRIPTION

Creates a checkpoint name for a block of Tcl commands, and evaluates that block of commands. When a block of Tcl is checkpointed, the user can then associate a set of actions or reports to run at that checkpoint.

The checkpoint name should be a Tcl string (without spaces) that is descriptive of the Tcl command steps being checkpointed. Every checkpoint name in the system must be unique. If a non-unique name is given, a number is automatically incremented and appended to the end until a unique name is achieved. For example: if a checkpoint named place_design already exists and a new eval_checkpoint is executed with the same checkpoint name, it will use the name place_design2 instead.

The Tcl command list that is executed by the checkpoint can be any valid Tcl. This includes a single Tcl command (with or without options), a block of multiple Tcl commands, or no Tcl commands (either exclude the command_list option or provide an empty Tcl string with "" or {}).

If an eval_checkpoint statement is nested inside another eval_checkpoint statement, then the inner one will be ignored. The enclosed Tcl commands for the inner eval_checkpoint will be evaluated as though no eval_checkpoint statement was present.

When an eval_checkpoint command is evaluated, the following sequence of steps occur in order:

- Runs any reports specified to run before the checkpoint using associate_checkpoint_report -before
- Runs any actions specified to run before the checkpoint using associate_checkpoint_action -before
- Runs the checkpointed block of Tcl commands. Alternately, this is skipped and any actions specified to run as a replacement for the checkpointed Tcl using associate_checkpoint_action -replace are run.
- Runs any actions specified to run after the checkpoint using associate_checkpoint_action -after
- Runs any reports specified to run after the checkpoint using associate_checkpoint_report -after

EXAMPLES

Creates a checkpoint named multibit and evaluates the identify_multibit command.

```
prompt> eval_checkpoint multibit { identify_multibit -register -apply }
```

Creates a checkpoint called placement_constraints and evaluates multiple Tcl commands.

```
prompt> eval_checkpoint placement_constraints {
  source ./scripts/placement_app_options.tcl
  source ./scripts/placement_blockages.tcl
  source ./scripts/placement_bounds.tcl
}
```

Creates a checkpoint named route_final and evaluates no Tcl. This checkpoint only exists as a reference point to associate with ac

```
prompt> eval_checkpoint route_final {}
```

SEE ALSO

create_checkpoint_action(2)
remove_checkpoint_actions(2)
create_checkpoint_report(2)
remove_checkpoint_reports(2)
associate_checkpoint_action(2)
associate_checkpoint_report(2)
get_current_checkpoint(2)
set_checkpoint_options(2)
reset_checkpoints(2)
get_checkpoint_data(2)

eval_pg_ml_model

Evaluate the quality of the trained PG ML model for fixability prediction.

SYNTAX

status **eval_pg_ml_model**

Data Types

ARGUMENTS

DESCRIPTION

This command evaluates the quality of the trained PG ML model by outputing the following ML prediction results: true_negative, false_negative, true_positive, and false_positive based on ML model and ML data in memory.

For example,

---------- testing report ---------- Unfixable: 1078 , Fixable: 346 True Negative: 1078, Ratio: 100.00% False Negative: 0, Ratio: 0.00% True Positive: 341, Ratio: 98.55% False Positive: 5, Ratio: 1.45%

EXAMPLES

The following example evaluates PG ML model stored in the directory ./pg_model with respect to ML data in memory.

```
prompt> compile_pg -create_ml_data
prompt> eval_pg_ml_model
```

SEE ALSO

compile_pg(2)
train_pg_ml_model(2)

create_pg_vias(2)

eval_with_undo

Executes a command or set of commands with undo-related settings applied.

SYNTAX

string **eval_with_undo**
[-atomic *name*]
[-disable]
commands

Data Types

name string
commands string

ARGUMENTS

-atomic *name*

Make the execution of *commands* appear on the undo command stack as though it were a single command level with name *name*.

-disable

Disable the recording of undo history during the execution of *commands*.

commands

The Tcl command or set of commands to execute with the applied undo settings.

DESCRIPTION

This command executes a set of Tcl commands as a single unit, with the option of disabling undoability of the commands in *commands*. If you use the **-atomic** option, the execution of the commands in *commands* appears as a single command with name *name* in the undo stack, and undoing *name* restores the database to the state before the execution of this command.

If you use the **-disable** option, undo is disabled for the duration of the command. It has the side effect of clearing the undo history. It is used in situations where it is known a priori that undo will not be necessary, and hence eliminates the runtime and memory overhead necessary to build and store undo data for the command. This option is most useful in cases where the magnitude of the expected changes is large. This command with the **-disable** option is the equivalent to the following sequence:

```
set undo [get_app_option_value -name shell.undo.enabled]
set_app_options -name shell.undo.enabled -value false
eval {commands}
if { $undo == "true" } { set_app_options -name shell.undo.enabled -value true }
```

The return value is that returned by the last Tcl command in *commands*.

EXAMPLES

The following example executes link_design with undo temporarily disabled:

```
prompt> eval_with_undo -disable { link_design }
```

SEE ALSO

undo(2)
redo(2)
create_undo_marker(2)
get_undo_info(2)

exec_gds2rh

Generates a Redhawk macro model using the RedHawk *gds2rh* utility.

SYNTAX

status **exec_gds2rh**
-config_file *config_file*

Data Types

string *config_file*

ARGUMENTS

-config_file *config_file*

Specifies a configuration file for RedHawk *gds2rh* utility.

DESCRIPTION

This command generates a Redhawk macro model by using RedHawk *gds2rh* utility with user-provided configuration file. This command does not check if there is any syntax error in the user configuration file.

EXAMPLES

The following example generates a RedHawk macro model using the user-provided configuration file as an input.

```
prompt> set_app_options -rail.redhawk_path -value /install_path/bin
prompt> exec_gds2rh user.config
```

SEE ALSO

analyze_rail(2)
set_app_options(2)

exit

Terminates the application.

SYNTAX

integer **exit**
[*exit_code*]

Data Types

exit_code integer

ARGUMENTS

exit_code

Specifies the return code to the operating system. The default value is 0.

DESCRIPTION

This command exits from the application. You have the option to specify a code to return to the operating system.

If there are pending jobs from **redirect -bg, then exit will wait for the jobs to complete before exiting.**

EXAMPLES

The following example exits the current session and returns the code 5 to the operating system. At a UNIX operating system prompt, verify (**echo**) the return code as shown.

```
prompt> exit 5

% echo $status
5
```

SEE ALSO

quit(2)

expand_objects

Expands the specified objects by moving each side out until it touches another object or the core boundary; in this case the number of sides is retained. Alternatively, the command expands the object to fill available space around it; in this case the number of sides is not retained.

SYNTAX

status **expand_objects**
[*object_list*]
[-side left | right | top | bottom | all]
[-offset *offset*]

Data Types

object_list collection
offset float

ARGUMENTS

object_list

Collection or selection set containing objects to expand. If this option is not specified, the global selection set is used.

-side left | **right** | **top** | **bottom** | **all**

Specifies the side of the object to expand. Use multiple **expand_objects** commands to expand the object on more than one side. The default is all, which expands the object to all sides.

-offset *offset*

Specifies the spacing between the sides and the surrounding objects. The offset value can be a negative or positive float value. The default is 0.0.

DESCRIPTION

This command expands an object's sides until they touch another object or the core boundary.

The expansion is done by moving each specified object side outwards until it hits one the specified object types. The types of objects considered for collisions can specified by **set_edit_settings** command.

If an offset is specified then the resultant shape's selected sides are shrunk (or grown in the case of negative offsets) by this offset after expansion.

If the operation results in an invalid object boundary (self overlapping) the command will fail.

Snapping is done automatically using global snap settings.

EXAMPLES

The following example resizes the alu1 cell so that it's bounding box is 20 units from any of the collision objects in all directions.

prompt> **expand_objects -side all -offset 10**

The alternative syntax uses collection to specify objects.

prompt> **expand_objects [get_selection] -side all -offset 10**

SEE ALSO

change_selection(2)
get_edit_setting(2)
get_selection(2)
set_edit_setting(2)

expand_outline

Expands the outline representation for a design into the full design representation. The command reads the original Verilog netlist to perform the expansion.

SYNTAX

string **expand_outline**
[-design *design_name*]
[-force]
[-strip *prefix_list*]
[-keep]
[-no_def]

Data Types

design_name string
prefix_list list

ARGUMENTS

-design *design_name*

Specifies the name of the design to expand.

-force

Forces the tool to reread the Verilog file.

-keep

Keeps the outline design open after expansion.

-no_def

Prevents reading of the DEF file during design expansion.

-strip *prefix_list*

Specifies a list of path names to remove from file names when searching.

DESCRIPTION

The **expand_outline** command rereads the Verilog files containing the modules for an outline design and generates a full design representation. The tool reads only the parts of the Verilog files containing the modules in the specified outline design.

The original Verilog files must be in the same location as when the **read_verilog_outline** command was run. If the files are not found in their original location, the tool searches for them by using the *search_path*. The **-strip** option removes file path name prefixes when searching with the *search_path*. If the files are modified, the tool skips them unless you specify the **-force** option.

After successful expansion, the tool closes the outline design representation. If the outline design is not referenced or modified, it is removed from memory. Specify the **-keep** option to retain both the outline design and the expanded design in memory.

Use the **-force** option with caution. The modules in each Verilog file are referenced by their original line numbers assigned by the **read_verilog_outline** command. If the line number location of a module changes for any reason, such as inserting a comment or deleting a blank line, the **expand_outline** command will fail to expand the module.

The **expand_outline** command behaves in a similar way to the **read_verilog** command. See the **read_verilog** command documentation for more information on options and variable settings.

EXAMPLES

In the following example, the **read_verilog_outline** command reads the r4000.v compressed Verilog netlist and creates an outline design. The **commit_block** command creates a separate design for the alu. The **expand_outline** command rereads the Verilog netlist and expands the top design.

```
prompt> set_ref_libs -ref_libs {"tcbn90ghvt" "tcbn90glvt"}
tcbn90ghvt tcbn90glvt
prompt> read_verilog_outline -top r4000 r4000.v.gz
Loading verilog file '/usr/.../r4000.v.gz'
1
prompt> current_design r4000
{"r4000"}
prompt> link
Information: units loaded from library 'tcbn90ghvt' (LNK-040)
Design 'r4000' was successfully linked.
1
prompt> commit_block -design mAlu alu
1
prompt> expand_outline
Loading verilog file '/tmp/r4000.hcvr.v.gz'
1
prompt>
```

SEE ALSO

commit_block(2)
expand_outline(2)
read_verilog(2)
set_cell_hierarchy_type(2)
outline_attributes(3)

shell.common.tmp_dir_path(3)
search_path(3)

explore_logic_hierarchy

Creates or removes module boundaries in the GUI to help analyze the logical hierarchy and prepare for committing blocks. The command converts all top-level logical hierarchies, or only specified hierarchies, and displays the result in the GUI. Use this command to determine which logical hierarchies should be converted to physical hierarchies by analyzing the module boundaries sizes, macro contents, and connectivity. For a flattened design, use **set_app_options -list { plan.place.hierarchy_by_name true}** to extract the name-based virtual hierarchies and create the corresponding move bounds of the virtual hierarchies for exploration.

SYNTAX

collection **explore_logic_hierarchy**
[-cell *cell_names*]
[-threshold *percent_value*]
[-utilization *utilization_value*]
[-virtual_group *module_boundary_names*]
[-name *new_module_boundary_name*]
[-virtual_ungroup *module_name*]
[-output *file_name*]
[-create_module_boundary |
-expand |
-collapse |
-remove |
-organize |
-force |
-place |
-use_existing_placement |
-rectangular]
[-nested]

Data Types

cell_names string
percent_value float
utilization_value float
module_boundary_names collection
new_module_boundary_name string
module_name string
file_name string

ARGUMENTS

-cell *cell_names*

Specifies the cell names of the logical hierarchies to process. You must specify one or more valid cell names together with the **-expand** or **-collapse** option. By default, the command processes all logical hierarchies at the top level. For a flattened design with

set_app_options -list { plan.place.hierarchy_by_name true}, specify virtual hierarchy names to be explored.

-threshold *percent_value*

Specifies the minimum block size to process. This option is valid only when option **-create_module_boundary** or option **-expand** is specified. If the *percent_value* is less than zero, the tool issues a warning message and sets the value to zero. By default, the *percent_value* is 0.03 and any logical hierarchy whose size is smaller than 3 percent of total size is ignored with option **-create_module_boundary**, and any child hierarchy whose size is smaller than 3 percent of the size of current expanding hierarchy is ignored with option **-expand**.

The default value of 0.03 can be changed using the global application option **plan.explore.default_threshold**.

If the **-cell** option is specified together with option **-create_module_boundary**, the **-threshold** option is ignored as the cells to be processed have been explicitly specified.

-utilization *utilization_value*

Specifies the utilization value when creating the boundary for a logic hierarchy. If the specified value is less than zero, this option is ignored and the design utilization is used. Valid utilization values are between 0.01 and 10. By default, the design utilization value is used. This option only works with the **-create_module_boundary** and **-expand** options.

-virtual_group *module_boundary_names*

Creates a module boundary by combining the specified module boundaries. The module boundaries specified in *module_boundary_names* are not required to belong to a single logical hierarchy. This option needs to be used together with the **-name** option. For the flattened design with **set_app_options -list { plan.place.hierarchy_by_name true}**, the existing virtual hierarchy move bounds to be grouped are specified.

-name *new_module_boundary_name*

Specifies the name of the new module boundary to be created. This option can only be used together with the **-virtual_group** option.

-virtual_ungroup *module_name*

Specifies the name of a grouped module boundary to ungroup.

-output *file_name*

Writes a file that contains the module boundary names and virtual group names.

-create_module_boundary

Creates a module boundary for the input hierarchy specified by the **-cell** option. If the **-cell** option is omitted, the **-create_module_boundary** option removes all module boundaries and re-creates the module boundaries for all top-level logic hierarchies, based on the **-threshold** option setting.

-expand

Expands an existing module boundary into subhierarchies. The command removes the existing module boundary for the specified hierarchy and creates a new hierarchical boundary for each subhierarchy. The tool creates new module boundaries for subhierarchies with a size larger than the threshold. The input logical hierarchy must be a module boundary, otherwise it cannot be expanded.

The tool removes the original module boundary and creates the new hierarchical boundaries on top of the original location. The new boundaries might overlap other existing module boundaries. The expanded module boundaries are selected in GUI; you can hand edit the locations to remove overlaps.

-collapse

Collapses the input hierarchy into its parent hierarchy. The tool removes all child module boundaries of the parent hierarchy and

creates a new module boundary for the parent.

-remove

Removes one or more module boundaries. If you specify **-cell cell_names**, the tool removes only the specified boundaries. If you do not specify the **-cell** option, the tool removes all module boundaries. The tool places macros from the module boundary to the upper-right of the design boundary.

-organize

Rearranges the existing module boundaries, top-level macros, and pads of which the physical status is unplaced into three predefined regions. After experimenting with different module boundary placements, you can use this option to set the module boundary to rectangular and reset the placements.

-force

Forces the tool to expand the hierarchy, even if it has no connectivity.

-place

Shows a placement of the module boundaries into the floorplan boundary. The module boundaries with "fixed" attribute and the blocks (committed cells) are treated as fixed (i.e., location is not changed) when other module boundaries are placed.

-use_existing_placement

Places module boundaries based on their existing placement, rather than using the result from a new placement run. This option is valid only when used with the **-place** option.

-rectangular

Restricts the shape of the module boundaries to be rectangular shapes only. Use this option together with the **-place** option. If you specify the **-place** option without the **-rectangular** option, the tool can create nonrectangular module boundaries to reduce the overlap between boundaries.

-nested

By default nested module boundaries are not allowed. For example, you cannot use "-create_module_boundary" to create module boundaries for hierarchy A and hierarchy A/B at the same time. With this option, nested module boundaries are allowed. Behavior of other options like "-expand", "-collapse", and "-place" are changed to create a nested result.

DESCRIPTION

The **explore_logic_hierarchy** command creates, removes, and places module boundaries to enable hierarchy and floorplan exploration in the GUI. You can create module boundaries for all top-level logical hierarchies, or select specific hierarchies by using the **-cell** option. Use this command to determine which logical hierarchies should be converted to physical hierarchies based on the size of the module boundaries, number of macros, and connectivity.

You can specify only one of the **-expand**, **-collapse**, **-remove**, and **-create_module_boundary** options.

This command returns TCL_OK on success with a collection of created objects, TCL_Error otherwise.

EXAMPLES

The following example creates module boundaries for all top-level logical hierarchies with a cell size larger than 1 percent of the total cell size. This is normally the first exploration option to apply.

```
prompt> explore_logic_hierarchy -create_module_boundary -threshold 0.01
```

The following example expands the input logical hierarchy named "I_ORCA_TOP".

```
prompt> explore_logic_hierarchy -cell I_ORCA_TOP -expand
```

The following example collapses the input logical hierarchy "I_ORCA_TOP/I_PCI_TOP" into its parent "I_ORCA_TOP".

```
prompt> explore_logic_hierarchy -cell I_ORCA_TOP/I_PCI_TOP -collapse
```

The following example removes the module boundary for "I_ORCA_TOP/I_PCI_TOP".

```
prompt> explore_logic_hierarchy -cell I_ORCA_TOP/I_PCI_TOP -remove
```

The following example places the module boundaries of the design.

```
prompt> explore_logic_hierarchy -place
```

SEE ALSO

initialize_floorplan(2)

export_advanced_technology_rules

Exports advanced technology placement rules from the current design for consumption by PrimeTime-ECO.

SYNTAX

status **export_advanced_technology_rules**
output_file
[-exclude_lib_cells *lib_cells*]
[-use_implant_layer_names]
[-attach]

Data Types

output_file string
lib_cells collection

ARGUMENTS

output_file

File name for output.

-exclude_lib_cells

Collection of library cells to exclude from color-based spacing rule processing. This option may be used to reduce command runtime if you know that certain library cells are not intended for use. By default, the command processes all library cells from all reference libraries attached to the current block.

-attach

Attaches the output of the command to the NDM design library for the design. This option does not affect the creation of the output file. By default, the command output is not attached to the NDM.

-use_implant_layer_names

Uses the VT implant layer names that correspond to the names in the technology file. By default, the command writes out abstracted layer names.

DESCRIPTION

This command exports all advanced technology placement rules, as well as all inter-cell spacing rules that have been set by the user, in the current design. The output of the command is encrypted. You can optionally choose to attach the output of the command

to the NDM of the current design as well.

SEE ALSO

report_placement_spacing_rules(2)
set_placement_spacing_label(2)
set_placement_spacing_rule(2)

extract_model

Creates Extracted Timing Model (ETM) library for the current block.

SYNTAX

status **extract_model**
[-lib_name *lib_name*]
[-etm_lib_work_dir *dir_name*]

Data Types

lib_name string
dir_name string

ARGUMENTS

-lib_name *lib_name*

Specifies the ETM library name. By default, the tool will generate ETM library with the same name as the top module name of the current block.

-etm_lib_work_dir *dir_name*

Specifies the directory where ETM library creation files and logs are generated. If you don't specify a working directory, a directory with the name ETM_Lib_work_dir is created under your current working directory.

DESCRIPTION

This command generates Extracted Timing Model (ETM) Library.

When the extract_model command executes, the following actions happen:

- Generates PrimeTime setup based on IC Compiler II multi scenario
- Invokes Distributed Multi Scenario Analysis (DMSA) PrimeTime under the hood unless PrimeTime has started already
- Runs multi scenario timing updates
- Generates Extracted Timing Model (ETM). If you want to influence ETM generation using any PrimeTime extract_model* variables, include the same in a file and provide the file as an argument to set_pt_options -post_link_script option.
- Invokes IC Compiler II Library Manager (icc2_lm_shell) to assemble ETM.db with the frame view to create ETM library. By

default, the tool will write out the generated ETM library into [pwd]/ETM_Lib_work_dir/block_name/label_name/ directory for labelled blocks and [pwd]/ETM_Lib_work_dir/block_name/ for non-labelled blocks. If you specify a value to etm_lib_work_dir, the tool will then write into [pwd]/<etm_lib_work_dir>/block_name/label_name/ for labelled blocks and [pwd]/etm_lib_work_dir/block_name/ for non-labelled blocks.

Multi-Level Hierarchy Support To generate an ETM library at the parent level, the child blocks must be linked to design views. In addition, full chip timing constraints should be provided using set_pt_options -scenario_constraints.

EXAMPLES

The following example script shows how to use extract_model to generate ETM library. The ETM library name would be the top module name of block block.design which would be created in [pwd]/ETM_Lib_work_dir/block/

```
icc2_shell> open block.nlib:block.design
icc2_shell> create_frame <options>  ;# extract_model needs Frame for library preparation
icc2_shell> set_pt_options -pt_exec_path <> \
        -post_link_script <tcl script with extract_model* variables> \
        -work_dir ETM_work_dir
icc2_shell> extract_model
```

The following example shows the creation of ETM library with user specified name using -lib_name. The ETM library my_etm_lib would be created in [pwd]/ETM_Lib_work_dir/block/test/

```
icc2_shell> open block.nlib:block/test.design
icc2_shell> create_frame <options> ;# extract_model needs Frame for library preparation
icc2_shell> set_pt_options -pt_exec_path <> \
        -post_link_script <tcl script with extract_model* variables> \
        -work_dir ETM_work_dir
icc2_shell> extract_model -lib_name my_etm_lib
```

The following example shows how to redirect ETM library creation data using -etm_lib_work_dir option. The ETM library my_etm_lib would be created in [pwd]/<my_etm_work_dir>/block/test/

```
icc2_shell> open block.nlib:block/test.design
icc2_shell> create_frame <options> ;# extract_model needs Frame for library preparation
icc2_shell> set_pt_options -pt_exec_path <> \
        -post_link_script <tcl script with extract_model* variables> \
        -work_dir ETM_work_dir
icc2_shell> extract_model -lib_name my_etm_lib -etm_lib_work_dir my_etm_work_dir
```

SEE ALSO

set_pt_options(2)
report_pt_options(2)
set_starrc_options(2)
report_starrc_options(2)

extract_svf

Extracts checkpoints from the specified .svf file.

SYNTAX

status **extract_svf**
[-dir *directory*]
[-ckpt *checkpoints*]
[-list]
file_name

Data Types

file_name string
directory string
checkpoints list

ARGUMENTS

-dir

Specifies the directory to save the extracted checkpoints. This option is mutually exclusive with the **-list** option.

-ckpt

Specifies a list of checkpoints to extract. If this option is not specified, all checkpoints will be extracted.

-list

Lists available checkpoints with the specified *file_name*. This option is mutually exclusive with the **-dir** option.

file_name

Specifies the .svf file that contains the checkpoints to extract.

DESCRIPTION

This command extracts checkpoints within the .svf file if there are any.

EXAMPLES

In the following example, the **extract_svf** command extracts the ckpt_logic_opt checkpoints and saves them to the directory named output.

```
prompt> extract_svf -ckpt "ckpt_logic_opt" -dir output my.svf
1
```

SEE ALSO

set_svf(2)

filter_collection

Filters an existing collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

collection **filter_collection**
[-regexp]
[-nocase]
collection1
expression

Data Types

collection1	collection
expression	string

ARGUMENTS

-regexp

Specifies that the =~ and !~ filter operators will use real regular expressions. By default, the =~ and !~ filter operators use simple wildcard pattern matching with the * and ? wildcards.

-nocase

Makes the pattern match case-insensitive.

collection1

Specifies the base collection to be filtered. This collection is copied to the result collection. Objects are removed from the result collection if they are evaluated as **false** by the conditional *expression* value. Substitute the collection you want for *collection1*.

expression

Specifies an expression with which to filter *collection1*. Substitute the string you want for *expression*.

DESCRIPTION

Filters an existing collection, resulting in a new collection. The base collection remains unchanged. In many cases, application commands that create collections support a *-filter* option that filters as part of the collection process, rather than after the collection has been made.

This type of filtering is almost always more efficient than using the **filter_collection** command after a collection has been formed. The **filter_collection** command is most useful if you plan to filter the same large collection many times using different criteria.

The **filter_collection** command results in either a new collection or an empty string. A resulting new collection contains the subset of the objects in the input *collection1*. A resulting empty string (the empty collection) indicates that the *expression* filtered out all elements of the input *collection1*.

The basic form of the conditional expression is a series of relations joined together with AND and OR operators. Parentheses () are also supported. The basic relation compares an attribute name with a value through a relational operator.

For example,

```
is_hierarchical == true and area <= 6
```

If an attribute is a collection attribute that contains a single object, then you can query an attribute value from that object using "." to name the attribute on the other object. The "." operator can be chained. For example.

```
owner.is_hierarchical == true and owner.area <= 6
shape.net.name==reset
```

The value side of a relation can be a simple string, quoted string, or an attribute name prefixed with "@". For example:

```
input_delay<=@output_delay
input_delay<=0.24
name=="@literal_name"
```

The relational operators are

```
==   Equal
!=   Not equal
>    Greater than
<    Less than
>=   Greater than or equal to
<=   Less than or equal to
=~   Matches pattern
!~   Does not match pattern
```

The basic relational rules are

- String attributes can be compared with any operator.
- Numeric attributes cannot be compared with pattern match operators.
- Boolean attributes can be compared only with == and !=. The value can be only either *true* or *false*.

Existence relations determine if an attribute is defined or not defined for the object. For example,

```
(sense == setup_clk_rise) and defined(sdf_cond)
```

The existence operators are

```
defined
undefined
!defined
```

These operators apply to any attribute as long as it is valid for the object class.

Collection attributes support a special sizeof(attrName) operator that returns the number of objects in the attribute. If the attribute is not set then 0 is returned.

The **filter_collection** command has a *-regexp* option that uses regular expressions when matching for string attributes. Regular

expression matching is done in the same way as in the Tcl **regexp** command. When using the *-regexp* option, take care in the way you quote the filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended.

Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed. You can make the regular expression search case-insensitive by using the *-nocase* option.

If the application variable filter_collection_extended_syntax is true then mathematical expressions are supported too. All function names must be prefixed with the '#' in order to disambiguate the function names from subscripted attribue names (i.e. "#sin(attr)"). In order to use a math expression on the right hand side of a comparison, the expression must be contained in parenthesis. This is required for backward compatibility with the legacy expression syntax as strings on the right do not require quoting.

When extended syntax is on filters can also match against glob style subscript patterns (i.e. "input_delay(*)<1.0"). In that case the term evaluates to true if any subscript satisfies the condition.

Additionally when extended syntax is on, a filter may refer to nested attributes that may imply multiple objects in a collection (i.e. "pins.capacitance==0.2"). In that case the term returns true if any of the objects referred to in the have an attribute value satisfying the condition.

EXAMPLES

The following example from PrimeTime creates a collection of only hierarchical cells.

```
pt_shell> set a [filter_collection [get_cells *] \
"is_hierarchical == true"]
{"Adder1" "Adder2"}
```

The following example from PrimeTime uses existence operators to create a collection of all nonmoded cell timing_arc objects in the current design.

```
pt_shell> set b [filter_collection \
        [get_timing_arcs -of_objects [get_cells *]] \
        "undefined(mode)"]
```

The following example finds all cells from {U0(1) U1(1) U0(0) U1(0)}
with the index (0). Notice the round brackets used for the index.

```
pt_shell> filter_collection -regexp [get_cells] {full_name =~ "U.*\(0\)"}
{U0(0) U1(0)}
```

The following example finds all ports from in[18] to in[23]. The placement of double quotes and backslashes is necessary to avoid syntax errors.

```
pt_shell> filter_collection [get_ports] -regexp {name=~ "in\[(1[8-9]|2[0-3])\]"}
{in[23] in[22] in[21] in[20] in[19] in[18]}
```

The following example finds the U2 and U4 cells, using the -nocase option:

```
pt_shell> filter_collection [get_cells] -regexp -nocase {name =~ u[24]}
{U2 U4}
```

SEE ALSO

collections(2)
regexp(2)
filter_collection_extended_syntax(3)

find_objects

Finds logical hierarchy objects within a scope. This is a UPF query command.

SYNTAX

list **find_objects**
scope
-pattern *pattern_string*
[-object_type inst | port | net | model | supply_port | process]
[-direction in | out | inout]
[-transitive true | false]
[-traverse_macros]
[-non_leaf]
[-leaf_only]
[-exact]
[-regexp]
[-ignore_case]

Data Types

scope string
pattern_string string

ARGUMENTS

scope

Specifies the scope within which to search for logical hierarchy objects.

-pattern *pattern_string*

Specifies a search pattern. By default, the pattern is treated as a Tcl glob expression.

-object_type inst | port | net | model | supply_port | process

Searches only for the specified object type.

The keyword values have the following definitions:

- **inst** (cell instance)
- **port** (design port)
- **net** (net)
- **model** (model or lib cell)

- **supply_port** (supply port)
- **process** (named process)

If **-object_type model** is specified, **find_objects** searches for instances of any model or lib cell whose name matches the *pattern_string*.

If you do not specify the **-object_type** option, instances, named processes, ports, and nets are searched.

-direction in | **out** | **inout**

Restricts the direction of the searched ports and supply ports.

This option has an effect only when ports and supply ports are included in the search.

-transitive true | **false**

Controls whether the descendants of the elements are included in the search.

By default (**false**), the descendants are not included in the search.

-traverse_macros

Controls whether the command would descend and traverse all hierarchies without stopping at any boundary marked as "soft_macro" or "terminal_boundary" by **set_design_attribute**.

If **-traverse_macros** is specified, then **-transitive TRUE** is implicitly specified.

By default, the searching will be stopped at any boundary marked as "soft_macro" or "terminal_boundary".

-non_leaf

Limits the search to only nonleaf design elements (elements that have child elements).

If you do not specify the **-non_leaf** or **-leaf_only** option, both leaf and non-leaf design elements are searched.

-leaf_only

Limits the search to only leaf-level design elements (elements without child elements).

If you do not specify the **-non_leaf** or **-leaf_only** option, both leaf and non-leaf design elements are searched.

-exact

Disables the default Tcl glob pattern matching. For use when searching for objects that contain the * and ? wildcard characters as part of their names.

The **-regexp** option is mutually exclusive with the **-exact** option

-regexp

Option to interpret and match the given pattern as a regular expression rather than a Tcl glob expression

The **-regexp** option is mutually exclusive with the **-exact** option

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *pattern*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name.

Note that the use of the "/" to match the hierarchy separator is only allowed with "glob" type matching; it is not allowed with **-regexp**. Even though the "/" is not treated as hierarchical separator in regular expression mode, it will be treated as normal alphabet to compare base name. In the EXAMPLES section you can find an example to show this.

-ignore_case

Performs case insensitive searches. By default, all matches are case sensitive.

DESCRIPTION

The **find_objects** command finds logical hierarchy objects within a scope. It returns a list of the found hierarchical names relative to the active scope; when nothing is found, it returns a null string.

This is a UPF query command. The command would use of the "/" to match the hierarchy separator with "glob" type matching, and this behavior is different from other netlist query command, like **get_cells**, **get_pins**, etc.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

Here are some examples of usage of the command.

```
prompt> find_objects a -pattern * -object_type inst \
  -transitive TRUE -non_leaf
a/b1 a/b2
```

Suppose we have "bot" module/lib_cell instantiated in the design with instances "a/b1", "a/b2", and "b1". With find_objects we can see following results.

```
prompt> find_objects . -object_type model -pattern bot -non_leaf \
 -transitive TRUE
a/b1 a/b2 b1
```

Suppose we have a design which has instances "a1", "x1", "a2/b2/c" and "x2". "x1" has "a1", "a1" has "b1", and "b1" has "c"; "x2" has "a2/b2/c". With find_objects, we can see the results as follows. With **-transitive true** we can get all leaf instance names in the design with pattern *a*/b*/c*.
Note that in case of transitive, the pattern should match the last string, that's to say, when we use pattern x1/a1/b1*, we can't get the instance x1/a1/b1/c:

```
prompt> find_objects . -pattern a*/b*/c -object_type inst -transitive true
a1/b1/c a2/b2/c x1/a1/b1/c x2/a2/b2/c
prompt> find_objects . -pattern a*/b*/c -object_type inst -transitive false
a1/b1/c a2/b2/c
prompt> find_objects . -pattern x1/a1/b1* -object_type inst -transitive true -leaf_only
prompt> find_objects . -pattern a2/b2/c -object_type inst -regexp
Warning: The use of the '/' to match the hierarchy separator is only allowed
with glob type matching; it is not allowed with -regexp. (UPF-366)
a2/b2/c
prompt> find_objects . -pattern a1/b1/c -object_type inst -regexp
Warning: The use of the '/' to match the hierarchy separator is only allowed
with glob type matching; it is not allowed with -regexp. (UPF-366)
```

Use the design in the previous example. Suppose we mark instance x2 as "terminal_boundary" using **set_design_attribute**. We can see the command won't return the instances below **x2** without **-traverse_macros**.

```
prompt> find_objects . -pattern a*/b*/c -object_type inst -transitive true
a1/b1/c a2/b2/c x1/a1/b1/c
prompt> find_objects . -pattern a*/b*/c -object_type inst -traverse_macros
a1/b1/c a2/b2/c x1/a1/b1/c x2/a2/b2/c
```

SEE ALSO

get_cells(2)
get_nets(2)
get_ports(2)
set_scope(2)

fix_floorplan_rules

Move macros or create routing or placement blockages to fix violations of related floorplan spacing, enclosure, halo, area and width rules, and output a Tcl file to record the changes.

SYNTAX

collection **fix_floorplan_rules**
[-derive_macro_keepout]
[-adjust_hard_macros]
[-derive_routing_blockages]
[-adjust_std_cell_areas]
[-file *file_name*]
[-prefix *blockage_prefix*]
[-objects *macros*]
[-bbox *bounding_box*]

Data Types

file_name string
blockage_prefix string
macros list
bounding_box rectangle

ARGUMENTS

-derive_macro_keepout

Fix hard macro and std_cell_area spacing & halo rule violations by deriving hard keepout margin on hard macro.

-adjust_hard_macros

Fix hard macro to hard macro spacing/length rule violation, and block boundary to hard macro enclosure/location rule violation.

-derive_routing_blockages

Fix hard macro and routing blockage enclosure/halo rule violation.

-adjust_std_cell_areas

Fix hard macro to std_cell_area spacing/length rule violation, std_cell_area to std_cell_area spacing rule violation, block boundary to std_cell_area enclosure/location rule violation, std_cell_area width rule violation, std_cell_area poly rule violation and std_cell_area area rule violation.

-file *file_name*

Name of tcl script to record changes.

-prefix ***blockage_prefix***

Name of prefix to automatically added blockages.

-objects ***macros***

List of macro instances for which we will fix violations. If provided, only the violations that related to specified macros will be fixed. Mutually exclusive with -bbox.

-bbox ***bounding_box***

Bounding box of macros to consider for fixing violations. If provided, only macros that are completely covered by the box will be selected, then only the violations that related to selected macros will be fixed. Mutually exclusive with -objects.

DESCRIPTION

After the floorplan rule checker is done, you can use this command to fix the following violations:

- Derive the macros' hard keepout margin to satisfy the macro to std_cell_area spacing/halo rule.
- Move macros to satisfy all grid-type spacing/length rules, and enclosure/location rules from the block boundary.
- Create routing blockages after macro placement to satisfy all routing blockage to hard macro enclosure and halo rules.
- Create placement blockages after macro placement to make the standard cell area satisfy all width rules, poly rules, area rules, spacing/length rules to hard macro, enclosure rules from the block boundary and spacing rules to standard cell area.

EXAMPLES

The following commands fix violations of floorplan rule_1 on hard macros to the block boundary.

```
prompt> set_floorplan_spacing_rule -name rule_1 \
  -from_object_types hard_macro -to_object_types block_boundary
prompt> fix_floorplan_rules -adjust_hard_macros
```

SEE ALSO

set_floorplan_spacing_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_width_rules(2)
set_floorplan_area_rules(2)
set_floorplan_length_rules(2)
set_floorplan_location_rules(2)
remove_floorplan_rules(2)
report_floorplan_rules(2)

fix_mv_design

Fix Multi-Voltage and PVT violations of buffers/inverters/diodes in the design

SYNTAX

fix_mv_design
[-buffer]
[-diode]
[-from *driver_list*]
[-cells *cell_list*]
[-lib_cells *lib_cell_list*]
[-minimize_dual_rail]
[-level_shifter]
[-report_only]
[-report_all]
[-output *output_file*]
[-html]
[-csv]
[-verbose]
[-no_split]

Data Types

driver_list list of {driver_pins or driver_nets}
cell_list list of {cells}
lib_cell_list list of {library cells}
output_file string

ARGUMENTS

-buffer

Specify **fix_mv_design** to work on buffer trees. In the current release, either -buffer or -diode has to be specified.

-diode

Specify **-diode** to work on all diodes in the netlist.

-from

Specify the buffer trees that **fix_mv_design -buffer** should work on. If input is a pin, it specifies the driver pin of a full or partial buffer tree. If input is a net, its driver pin will specify a full or partial buffer tree. Wild cards can be used to match pins and nets.

-cells

Specify a list of cells that **fix_mv_design** can only work on. **fix_mv_design** will not modify any other cells outside of this list.

If both **-cells** and **-from** are specified, **fix_mv_design -buffer** will only work on buffers and inverters from the specified cell list which are present in the buffer trees specified by **-from**. Other cells in the cell list will not be modified.

-lib_cells

Specify a list of single-rail and dual-rail buffer/inverter lib cells that **fix_mv_design -buffer** can use. The command will not use any lib cells that are not found in this list. Tool will keep existing lib cells used by buffers/inverters if they don't need to be fixed.

-minimize_dual_rail

By default, **fix_mv_design -buffer** will not swap dual-rail buffers/inverters with single-rail lib cells if it does not fix any isolation violations. But if **-minimize_dual_rail** option is specified, **fix_mv_design** will minimize dual-rail buffers/inverters whenever possible without introducing new isolation violations.

-level_shifter

By default, buffer trees stop at level shifters and enable level shifters. If **-level_shifter** option is specified, buffer trees will include level shifters and the command can modify input/output supplies of these level shifters. However, library cells of these level shifters will not be changed.

-report_only

Do not modify netlist but print report only.

-report_all

Display detailed information of all buffer trees no matter they have been modified by the command or not. The **-verbose** option must also be specified.

-output <*output_file*>

Print report into <*output_file*>.

-html

Print report in HTML format.

-csv

Print report in CSV (Comma-Separated Value) format.

-verbose

Display detailed information of each buffer tree modified by the command.

-no_split

Display text in table without splitting lines.

DESCRIPTION

The **fix_mv_design** command with **-buffer** option will perform the following changes in buffer trees if feasible:

- Fix isolation violations by changing buffer/inverter lib cells and their supply net connections. No isolation cells will be inserted.

- Fix voltage violations by changing buffer/inverter lib cells and their supply net connections. No level shifter cells will be inserted.
- Fix PVT, bias, set_target_library_subset, set_libcell_subset, and set_lib_cell_purpose violations by swapping buffer/inverter lib cells with legal ones.
- Fix MV-058 violations (which complain about the backup supply of a dual-rail buffer/inverter being connected to the power domain primary supply) by replacing dual-rail with single-rail buffer/inverter lib cells.
- Assign physical/logical feedthrough buffer/inverter to a correct power domain which is consistent with its voltage area.
- Minimize number of dual-rail buffer/inverter lib cells by swapping them with single-rail counterparts without introducing new voltage or isolation violations if **-minimize_dual_rail** option is specified.

If the original buffer/inverter lib cell has already been placed, **fix_mv_design** will put the new lib cell in the same location. But **fix_mv_design** does not guarantee that the new placement is legal or timing is met. Please run optimization and legalization commands again.

PG net connections to the original buffer/inverter will be removed if the lib cell is changed by **fix_mv_design**. Please run **connect_pg_net** to reconnect the PG nets.

If **-diode** is specified, fix_mv_design will check all diodes in the current netlist. If the diode sits in the voltage area which is not matching the diode's power domain, then power domain of its voltage area will be assigned to the diode. If **-report_only** is specified, fix_mv_design will only report that certain power domain will need to be assigned to diodes without making any actual power domain assignment.

EXAMPLES

The following example runs **fix_mv_design** on a buffer tree driven by input pin 'in' to fix isolation violation and minimize number of dual-rail buffers. It does not modify the netlist but only reports what would have been modified:

```
prompt> fix_mv_design -buffer -from {in} -minimize_dual_rail -report_only -verbose
****************************************
Report : fix_mv_design
        -buffer
        -minimize_dual_rail
        -report_only
        -verbose
Design : top
Version: Q-2019.12
Date   : Sat Dec 21 14:16:17 2019
****************************************


=====================
 Buffer Tree Summary
=====================
 Total Buffer Tree                : 1
 Fixed Buffer Tree                : 1
 Dual-Rail to Single-Rail Swap    : 3
 Single-Rail to Dual-Rail Swap    : 1
 Buffer Tree with Isolation Violation : 0

============================
 Buffer Tree driven by 'in'
============================

 Buffer Tree Data        : Before After
 ----------------------  ------ ------
```

```
Load                : 3     3
Single-Rail Buffer      : 1     3
Dual-Rail Buffer        : 4     2
Single-Rail Inverter    : 0     0
Dual-Rail Inverter      : 0     0
Has Isolation Violation : Yes   No

ID Cell Name          Cell Type       Original Supply     New Supply          Original Lib Cell    New Lib Cell
-- -------------------- --------------- ------------------- ------------------- -------------------- --------------------
00 in                 Driver          (VDD0, VSS)         (VDD0, VSS)         -                    -
01 MID1/MY_BUF        Buf (DR to SR)  (VDD0, VSS)         (VDD1, VSS)         LIB/DUAL_RAIL_BUF    LIB/SINGLE_RAIL_BUF
02 MID2/MY_BUF        Buf (DR)        (VDD0, VSS)         (VDD6, VSS)         LIB/DUAL_RAIL_BUF    LIB/DUAL_RAIL_BUF
xx out[2]             Load            (VDD6, VSS)         (VDD6, VSS)         -                    -
03 MID3/MY_BUF        Buf (SR to DR)  (VDD3, VSS) !       (VDD4, VSS)         LIB/SINGLE_RAIL_BUF  LIB/SINGLE_RAIL_BUF
04 MID4/MY_BUF        Buf (DR to SR)  (VDD0, VSS)         (VDD4, VSS)         LIB/DUAL_RAIL_BUF    LIB/SINGLE_RAIL_BUF
xx out[0]             Load            (VDD6, VSS)         (VDD6, VSS)         -                    -
05 MID5/MY_BUF        Buf (DR to SR)  (VDD0, VSS)         (VDD5, VSS)         LIB/DUAL_RAIL_BUF    LIB/SINGLE_RAIL_BUF
xx out[1]             Load            (VDD6, VSS)         (VDD6, VSS)         -                    -

 Fanout Tree
-------------

[00]------[01]----+-[03]----+-[05]------[xx]
                  |       +-[04]------[xx]
                  +-[02]------[xx]

prompt> fix_mv_design -diode -report_only -verbose
****************************************
Report : fix_mv_design
        -diode
        -verbose
        -report_only
Design : addmult_shell
Version: Q-2019.12-SP2-DEV
Date   : Thu Dec 19 09:14:35 2019
****************************************
Info: need to apply power domain ADDMULT_OFF on diode instance ADDMULTINST_OFF/ADDINST/ADDERINST/clk_gate_cout_
Info: 1 of diodes need to have power domain applied onto explicitly
```

SEE ALSO

add_buffer(2)
add_buffer_on_route(2)
check_bufferability(2)
check_mv_design(2)
create_buffer_trees(2)
create_mv_cells(2)
remove_buffer_trees(2)
report_buffer_trees(2)
set_lib_cell_purpose(2)
set_libcell_subset(2)
set_target_library_subset(2)

fix_pg_missing_vias

Creates PG vias to fix missing vias by given DRC error objects that are written by the **analyze_rail** command. If PG vias are inserted successfully for a missing via error object, the error object's status will be updated to fixed.

SYNTAX

status **fix_pg_missing_vias**
  error_objects
  -error_data *drc_error_data*
  [-via_masters *via_master_list*]
  [-allow_parallel_objects]
  [-drc no_check | check_but_no_fix]
  [-mark_as *usage_type*]
  [-tag *tag_name*]

Data Types

error_objects collection
fldrc_error_data collection
via_master_list list
usage_type string
tag_name string

ARGUMENTS

error_objects

A collection of error objects to insert PG vias.

-error_data *drc_error_data*

Specifies the error data where the error object collection is from.

-via_masters *via_master_list*

Specifies the list of via masters to be used during via creation. You can specify via masters with the following information:

- Contact code names defined in the technology file
- Via rule names defined by **set_pg_via_master_rule**
- Custom via definitions
- The **default** keyword

Given one intersection, the tool uses the following rules for via creation in an order from high to low:

1. Via rules
2. Specified custom via contact definitions
3. Specified contact codes in the technology file
4. Specified contact codes or custom via definitions

For example, the tool first tries feasible via rules for via creation. If all via rules do not apply to this intersection, the tool uses the specified custom via contact definitions for via creation, and so on. If **NIL** is not specified in the list, the default contact code is used to complete a stacked via or to replace a via with DRCs when applicable. By default, the first DRC-clean default contact code is used for via insertion for each intersection.

-allow_parallel_objects

Specifies that vias are created between parallel objects. By default, vias are only created between orthogonal objects.

-drc no_check | **check_but_no_fix**

Specifies the DRC option for PG via creation. The argument following -drc must be either *no_check* or *check_but_no_fix*. When *no_check* is specified, the command will not perform DRC check while creating vias. When *check_but_no_fix* is specified, the command performs DRC check and reports DRC errors without fixing DRC errors. By default, DRC is checked and fixed for the inserted PG vias.

-mark_as *usage_type*

Specifies the type of the created vias. The argument must be one of the following:

- **stripe**: Power strap
- **ring**: Power ring
- **lib_cell_pin_connect**: Standard cell connection
- **macro_pin_connect**: Macro connection
- **user_route**: User route
- **follow_pin**: Standard cell connection
- **shield_route**: Shield route

If you do not specify this option, the tool uses the shape used on the bottom layer shape for the created vias.

-tag *tag_name*

Specifies a tag name to be assigned to all new vias and shapes created by this command. This tag name is used as a user attribute for filtering collections at later design stages. By default, no tags are assigned.

DESCRIPTION

The **fix_pg_missing_vias** command creates PG vias by a given error object collection. This command only supports error objects which are generated by the **analyze_rail -check_missing_via** command. For each error object, this command invokes **create_pg_vias** to insert PG vias. If PG vias are inserted successfully, the error object's status will be updated to "fixed". Or the status will be "ignored".

EXAMPLES

The following example creates PG vias for all error objects.

```
prompt> set errdm [open_drc_error_data miss_via.VDD.err]
prompt> set errs [get_drc_errors -error_data $errdm]
prompt> fix_pg_missing_vias -error_data $errdm $errs
```

SEE ALSO

create_pg_vias(2)
analyze_rail(2)
open_drc_error_data(2)
get_drc_errors(2)

fix_placement_color_mask

Perform color mask swapping of violated cell instances from the current design for consumption by PrimeTime-ECO.

SYNTAX

status **fix_placement_color_mask**

DESCRIPTION

This command swaps the maskShift attribute of cell instances that violated color rule. This command is for consumption by PrimeTime-ECO in conjunction with the export_advanced_technology_rules command. This command does not displace cells.

SEE ALSO

export_advanced_technology_rules(2)

fix_signal_em

Fixes signal electromigration violations.

SYNTAX

fix_signal_em
[-nets *net_list*]

Data Types

net_list list

ARGUMENTS

-nets *net_list*

Specifies the list of net names for which to fix signal EM violation. By default, all nets with EM violations are fixed.

DESCRIPTION

The **fix_signal_em** command performs automatic fixing of signal (interconnect) electromigration violations for the current design or for specified nets.

The command invokes electromigration analysis for nets, which calculates the current on every segment of the net and compares the result with the current thresholds defined in the design library. EM fixing strategies are applied on the violations.

Before using the **fix_signal_em** command, make sure the design is detail routed and has signal electromigration constraints. Also, make sure the switching activity has been defined for all boundary nets.

By default, the command repairs all EM violations in the whole design, including clock nets and signal nets. To restrict fixing to specified nets, list those nets in the command. You can use the **get_nets** command to create a collection of nets on which to perform EM fixing.

This command uses information from current mode and corner.

Multicorner-Multimode Support

This command works only on the current scenario.

EXAMPLES

The following example runs signal electromigration fixing.

```
prompt> fix_signal_em
```

SEE ALSO

report_signal_em(2)

flip_objects

Flips one or more objects in the current design.

SYNTAX

status **flip_objects**
-x *x* | -y *y* | -flip x | y
[-anchor center | ll | lr | ul | ur]
[-group]
[-force]
[-simple]
[*object_list*]

Data Types

x	float
y	float
object_list	collection or selection

ARGUMENTS

-x *x*

Specifies, in user units, either the x-coordinate of the anchor point (if you do not specify the **-anchor** or **-group** option) or the x-offset from the x-coordinate (if you specify the **-anchor** or **-group** option).

The resultant x-coordinate specifies the position of the vertical line to flip the objects about.

-y *y*

Specifies, in user units, either the y-coordinate of the anchor point (if you do not specify the **-anchor** or **-group** option) or the y-offset from the y-coordinate (if you specify the **-anchor** or **-group** option).

The resultant y-coordinate specifies the position of the horizontal line to flip the objects about.

-flip x | **y**

Specifies the axis around which to flip the object. Use the **-anchor** option to specify a corner of the object around which to flip. By default, each object is flipped around its center point.

-anchor center | **ll** | **lr** | **ul** | **ur**

Specifies the anchor point for rotation.

The anchor values are:

center is center
ll is lower left
lr is lower right
ul is upper left
ur is upper right

The option cannot be specified together with the **-group** option.

-group

Specifies the center point of a group as an anchor for rotation.

The option cannot be specified together with the **-anchor** option.

object_list

Collection or selection set containing objects to rotate. If this option is not specified, the global selection set is used.

-force

Resizes locked or fixed objects. By default, such objects are not resized according to global edit settings. Edit settings can be accessed using **get_edit_setting**.

-simple

Disable snapping and editing constraints. By default, objects are snapped according to global snap settings. Snap settings can be accessed using **get_snap_setting**.

DESCRIPTION

The **flip_objects** command flips one or more objects in the current design.

Note that snapping is done automatically by using global snap settings.

Flipping at X

To flip objects about an x-coordinate, the x-values of the boundaries of each object are mirrored about the x-value of the vertical line specified by the **-x** and **-anchor** options. The object then has its boundary set to the new values.

Flipping at Y

To flip objects about a y-coordinate, the y-values of the boundaries of each object are mirrored about the y-value of the horizontal line specified by the **-y** and **-anchor** options. The object then has its boundary set to the new values.

EXAMPLES

The following example flips all selected objects around the y-axis using center as an anchor.

```
prompt> flip_objects -flip y
```

The alternative syntax uses collection to specify objects.

```
prompt> flip_objects [get_selection] -flip y
```

SEE ALSO

change_selection(2)
get_edit_setting(2)
get_selection(2)
get_snap_setting(2)
move_objects(2)
rotate_objects(2)
set_edit_setting(2)
set_snap_setting(2)
snap_objects(2)

foreach_in_collection

Iterates over the elements of a collection.

SYNTAX

string **foreach_in_collection**
itr_var
collections
body

Data Types

itr_var	string
collections	list
body	string

ARGUMENTS

itr_var

Specifies the name of the iterator variable.

collections

Specifies a list of collections over which to iterate.

body

Specifies a script to execute per iteration.

DESCRIPTION

The **foreach_in_collection** command is used to iterate over each element in a collection. You cannot use the Tcl-supplied **foreach** command to iterate over collections because the **foreach** command requires a list, and a collection is not a list. Also, using the **foreach** command on a collection causes the collection to be deleted.

The arguments for the **foreach_in_collection** command parallel those of the **foreach** command: an iterator variable, the collections over which to iterate, and the script to apply at each iteration. All arguments are required.

Note: The **foreach_in_collection** command does not allow a list of iterator variables.

During each iteration, the *itr_var* option is set to a collection of exactly one object. Any command that accepts *collections* as an

argument also accepts *itr_var* because they are of the same data type (collection).

You can nest the **foreach_in_collection** command within other control structures, including another **foreach_in_collection** command.

Note that if the body of the iteration is modifying the netlist, it is possible that all or part of the collection involved in the iteration will be deleted. The **foreach_in_collection** command is safe for such operations. If a command in the body causes the collection to be removed, at the next iteration, the iteration ends with a message indicating that the iteration ended prematurely.

An alternative to collection iteration is to use complex filtering to create a collection that includes only the desired elements, then apply one or more commands to that collection. If the order of operations does not matter, the following are equivalent. The first is an example without iterators.

```
set s [get_cells {U1/*}]
command1 $s
command2 $s
unset s
```

The following is a similar approach using the **foreach_in_collection** command:

```
foreach_in_collection itr [get_cells {U1/*}] {
  command1 $itr
  command2 $itr
}
```

For collections with large numbers of objects, the non-iterator version is more efficient, though both produce the same results if the commands are order-independent.

EXAMPLES

The following example from PrimeTime removes the wire load model from all hierarchical cells in the current instance.

```
pt_shell> foreach_in_collection itr [get_cells *] {
?         if {[get_attribute $itr is_hierarchical] == "true"} {
?           remove_wire_load_model $itr
?         }
?       }
Removing wire load model from cell 'i1'.
Removing wire load model from cell 'i2'.
```

SEE ALSO

collections(2)
foreach(2)

generate_hot_spots

Generates voltage drop hot spots by dividing the whole design into multiple grids within which voltage drop hot spots can be analyzed. All grids will be sorted by voltage drop values. Each grid will be assigned with an index number start from 0, where the smaller the number, the bigger voltage drop the grid has.

SYNTAX

status **generate_hot_spots**
[-net *net_name*]
[-percentage *percentage*]
[-site_row *site_row_height_multiplier*]
[-row *number_of_rows*]
[-column *number_of_columns*]

Data Types

net_name string
percentage float
site_row_height_multiplier integer
number_of_row integer
number_of_columns integer

ARGUMENTS

-net *net_name*

Specifies a power or ground net name on which hot spot analysis is performed.

-percentage *percentage*

Specifies a percentage threshold which is used to generate hot spot error cells. Default is 0.1 which means 10%.

-site_row *site_row_height_multiplier*

By default the whole design is divided into multiple grids by using the standard cell row height and vertical PG straps on the lowest metal layer. This option allows users to use multiple row heights to divide hot spot rows while keeping columns separated by vertical PG straps.

-row *number_of_rows*

Specifies the number of row to construct grid boxes.

-column *number_of_columns*

Specifies the number of columns to construct grid boxes.

DESCRIPTION

The **generate_hot_spots** command generates voltage drop hot spot analysis by dividing the whole design into multiple grid boxes from which voltage drop hot spots can be analyzed. This command is needed for the **report_hot_spots** command.

EXAMPLES

The following example generates hot spot grid boxes using the default standard cell row height and vertical PG straps. Users can open the error browser to browse those voltage drop error cells.

```
prompt> generate_hot_spots -net VDD
Information: 1015 of instance effective voltage drop error cells created
```

The following example divides the whole design to 100 by 200 grid boxes:

```
prompt> generate_hot_spots -net VDD -row 100 -column 200
Information: 2314 of instance effective voltage drop error cells created
```

SEE ALSO

report_hot_spots(2)
remove_hot_spots(2)

generate_mv_constraints

Derives MV constraints automatically.

SYNTAX

status **generate_mv_constraints**
-elements *objects*
[-no_isolation]
[-apply | -output *file_name*]

Data Types

objects list
file_name string

ARGUMENTS

-elements *objects*

Specifies the objects on which to generate MV constraints. The objects can be pins of the root cells of power domains or ports of the top-level design for the top power domain, or hierarchical instances.

-no_isolation

Optional argument which enables the tool to derive UPF *-no_isolation* isolation strategies based on electrical needs. At least one of the *-no_isolation* options must be used.

-apply

Optional argument which instructs the tool to apply the newly-derived strategies to the design in memory.

-output *file_name*

Optional argument which instructs the tool where to write out the newly-derived strategies. If the file specified already exists, it is overwritten.

DESCRIPTION

The generate_mv_constraint utility automatically generates *-no_isolation* strategies for power domain boundary pins where an isolation strategy has been specified, but is not needed based on the power states defined in the UPF. The command has no effect on domain boundary pins where isolation cells have already been inserted.

When the tool-derived *-no_isolation* strategies are applied on top of existing user strategies, it prevents unnecessary isolation cells from being inserted.

At least one of the *-output* or *-apply* options must be specified. If the *-apply* option is not used, you are expected to load the output UPF file specified in the *-output* option manually.

EXAMPLES

The following example generates no_isolation strategies and writes the output to a user-specified file:

```
prompt> generate_mv_constraints -elements {B1 B2} no_isolation -output no_iso.upf
```

The output file no_iso.upf looks as follows:

```
set derived_upf true
if ($derived_upf) {
set_isolation snps_no_iso__0 -domain PD_P1 -no_isolation \
  -elements {B1/in  B1/out}
set_isolation snps_no_iso__0 -domain B2/PD_P2 -no_isolation \
  -elements B2/in
}
set derived_upf false
```

SEE ALSO

load_upf(2)
create_mv_cells(2)

generate_net_pattern

Generate automatic resistance scaling factors with net pattern.

SYNTAX

status **generate_net_pattern**
-output <file_name>
-route_mode <mode>
[-apply]

Data Types

file_name string
mode string

ARGUMENTS

-output *file_name*

Specifies the name of the output file to which pattern based resistance scaling table for the current design is written.

-route_mode *mode*

Specifies the routing stage that you want to run net pattern generation. For faster run time and good accuracy, track assign is the default. If you'd like to use full detail route, you can use *"-route_mode detail"*. If you have run routing, you can use *"-route_mode skip"*.

-apply

Once this option is used, the net pattern based scaling resistane table will be applied immediately so the next virtual route extraction will be using the table to adjust the resistance.

DESCRIPTION

For designs using advanced nodes, high metal layers show much smaller resistance compared to lower metal layers. In preroute extraction, the tool uses a weighted approach to estimate preroute resistance because it does not know which metal layers a net will eventually go on during detail routing.

To improve resistance correlation, the tool uses a resistance scaling approach based on the wire length net pattern. The router tends to route long nets on upper layers, which typically means a smaller resistance unit for long nets, so virtual routing can be pessimistic for long nets. For short nets, the router uses lower metal layers, which means that virtual route estimation can be optimistic. In this scaling approach, the tool scales the net's virtual route estimation based on scaling factors for each net pattern.

The default pattern generation will run global route and track assign, and then run pattern generator to create the net pattern based resistance scaling table for all corners. You can use -apply option to apply the generated net pattern immediately for the next virtual route extraction and future steps or you can save the pattern table to a file called "file_name" if option *-output* is used. If you have a routed design (for example, a detailed routed design using route_auto), you can use *"-route_mode skip"* option so that we will start the pattern generation without going through the routing steps. If you dont use *"-route_mode skip"* option, routes will be removed after pattern generation.

In theory, you can use the above command anywhere in your flow. You can either just create a pattern for future use using *"-output"* option (which will create a tcl file that you can source later on), or you can generate the pattern and use it immediately to drive later flows. In practice, a better flow may be to run your first flow all the way to route_auto and then create the net pattern there and reuse it to drive your future preroute optimization like place_opt and clock_opt. The following is a sample flow:

```
place_opt
clock_opt
route_auto
generate_net_pattern -route_mode skip -output my_pattern_r.tcl
```

Then restart your place_opt flow using the generated pattern tcl file:

```
source my_pattern_r.tcl
place_opt
clock_opt
route_auto
route_opt
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example generates and writes the net pattern resistane scaling table to an output file.

```
prompt> generate_net_pattern -output pattern.tcl
```

SEE ALSO

all_corners(2)
set_extraction_options(2)
report_extraction_options(2)
set_parasitic_parameters(2)
report_parasitic_parameters(2)
update_timing(2)
read_parasitics(2)

generate_pg_script

Creates a configuration file or a Tcl script that includes the syntax for pattern-based PG network synthesis.

SYNTAX

status **generate_pg_script**
[-input *file_name*]
[-outut *file_name*]
[-template]

Data Types

file_name string

ARGUMENTS

-input *file_name*

Specifies the name of the PG configuration file. This is a required option when the **-template** option is not specified.

-output *file_name*

Specifies the name of the PG script file that is written out. If the option is not specified, the default script name is ppns_script.tcl.

-template

Creates a PG configuration file. When this option is specified, all other options are ignored.

DESCRIPTION

This command creates a configuration file or a Tcl script that includes the syntax for pattern-based PG network synthesis.

Two configurations are supported in the PG configuration file: StrapConfig and ViaConfig. See the following descriptions for details.

TargetConfig: Specifies which configurations are used to create the PG script. Without it, all the configurations in the configuration file are used to create the PG script.

I. StrapConfig:

1. Mesh (with category mesh) Required keywords: target_area, category, nets, layer. Optional keywords: tag, blockage, extension. Required keywords for layer: widths, direction, spacings, pitch. Optional keywords for layer: offset, alignTrack, trim.

2. Std Rail (with category rail) Required keywords: target_area, category, nets, layer. Optional keywords: tag, blockage. Optional

keywords for layer: widths.

3. Alignment straps for switch cells (with category switch_alignment). Required keywords: target_area, category, nets, layer. Optional keywords: tag, blockage, extension. Required keywords for layer: alignCell. Optional keywords for layer: widths, direction, alignPinLayer, alignWireNumber, alignTrack, trim, offset.

4. Alignment straps for physical cells (with category physical_cell_alignment). Required keywords: target_area, category, nets, layer. Optional keywords: tag, blockage, extension. Required keywords for layer: alignCell. Optional keywords for layer: widths, alignPinLayer, alignPinName, alignWireNumber, alignTrack, offset.

5. Macro connection for long pin (with category macros and pintype long_pin). Required keywords: target_area, category, nets, layer. Optional keywords: tag, blockage. Required keywords for layer: pintype, pitch, direction. Optional keywords for layer: widths, spacings, pinlayer.

5. Macro connection for scattered pin (with category macros and pintype scattered_pin). Required keywords: target_area, category, nets, layer. Optional keywords: tag, blockage. Required keywords for layer: pintype. Optional keywords for layer: widths, pinlayer.

II. ViaConfig:

1. Via between wires (with category wire_to_wire). Required keywords: category, nets. conditionally required keywords: upper_layer, lower_layer. Optional keywords: via_def, tag, allow_parallel.

2. Via between wires and pins (with category wire_to_pin) Required keywords: category, nets, pin_name. conditionally required keywords: upper_layer, lower_layer. Optional keywords: via_def, tag, allow_parallel.

III. Keyword syntax:

1. target_area: Supported setting: core, design_boundary, voltage_areas, polygon, macros, blocks, pg_regions. Please see the man page of the command set_pg_strategy for details.

2. category: Supported setting: mesh, rail, switch_alignment, physical_cell_alignment, macros, wire_to_wire, wire_to_pin.

3. nets: Supported setting: names of power/ground nets.

4. layer: Supported setting: names of routing layers.

5. blockage: Supported setting: please see the man page of the option -blockgae of the command set_pg_strategy for details

6. extension: Supported setting: please see the man page of the option -extension of the command set_pg_strategy for details

7. tag: Supported setting: any name.

8. widths: Supported setting: values (in micron).

9. direction: Supported setting: horizontal, vertical.

10. spacings: Supported setting: values (in micron), minimum, interleaving.

11. pitch: Supported setting: values (in micron).

12. offset: Supported setting: values (in micron).

13. alignTrack: Supported setting: true, false

14. trim: Supported setting: true, false

15. alignCell: Supported setting: master name of cell.

16. alignPinLayer: Supported setting: name of layer.

17. alignWireNumber: Supported setting: integer.

18. pintype: Supported setting: long_pin, scattered_pin.

19. pinlayer: Supported setting: name of layer.
20. upper_layer: Supported setting: name of layer.
21. lower_layer: Supported setting: name of layer.
22. via_def: Supported setting: name of viaDef or contact code.
23. allow_parallel: Supported setting: true, false.

EXAMPLES

The following example creates a PG configuration file:

```
prompt> generate_pg_script -template
```

The following example creates a PG script file named test.tcl from the dp_pgConfigExample.txt PG configuration file:

```
prompt> generate_pg_script -input dp_pgConfigExample.txt -output test.tcl
```

SEE ALSO

create_pg_mesh_pattern(2)
set_pg_strategy(2)
compile_pg(2)
set_host_options(2)
create_pg_std_cell_conn_pattern(2)
create_pg_special_pattern(2)
set_pg_strategy_via_rule(2)
create_pg_vias(2)

generate_rm

Generate the IC Compiler II Reference Methodology Scripts

SYNTAX

status **generate_rm**
-rm_script <version_name>
-node <node_name>
[-foundry *foundry_name*]
-output <output_dir>
[-synopsys_library]

Data Types

version_name string
node_name string
foundry_name string
output_dir string

ARGUMENTS

-rm_script *version_name*

Specifies the script version of the IC Compiler II Reference Methodology to generate.

The *"Full RM Version"* is the script version that gives users control to configure with optional features, their flat place and route flow.

The *"JumpStart Version"* is the starting point for new users that have a block ready for place_opt. It will run the default recommended flat place and route flow.

-node *node_name*

Specifies the technology node version of the scripts to generate. Supported technology nodes are "established, 20nm, 16nm, 14nm, 12nm, 10nm, 7nm, and 5nm". The user must specify one of the supported names.

-foundry *foundry_name*

Specifies the name of the foundry for the technology node if applicable.

-output *output_dir*

Specifies the output directory location for the reference scripts. By default the output directory is the current working directory.

-synopsys_library

Specifies to generate the Synopsys Logic Library configuration of the scripts when applicable.

DESCRIPTION

This command will generate the IC Compiler II Reference Methodology scripts in the configuration that the user specifies. The user must configure for the technology node as well as foundry name and Synopsys Logic Library version when applicable.

The perferred method of generating these scripts are through the GUI of this command that can be accessed through the *Help* menu dropdown and selected with *Generate RM Scripts*.

EXAMPLES

In the following example the generate_rm command will generate the JumpStart Version of the 20nm technology node scripts in the current working directory.

prompt> **generate_rm -node 20nm -rm_script {JumpStart Version} -output ./**
File ./ICC2-RM_Q-2019.12.tar.gz is generated. Please unpack the file to get IC Compiler II Reference Methodology scripts.

SEE ALSO

generate_sadp_tracks

Creates tracks based on self-aligned double patterning (SADP) track rules.

SYNTAX

collection **generate_sadp_tracks**
-rule_name *track_pattern_rule*
-layer *layer_name*
[-start *point*]
[-stop *point*]
[-count *number_of_repeats_using_rule*]
[-pitch *pitch_between_patterns*]
[-auto_fill intra_pattern | inter_pattern | all | none]
[-exact]
[-create_pg_route]
[-color mask_one | mask_two]

Data Types

track_pattern_rule string
layer_name string
point float
number_of_repeats_using_rule int
pitch_between_patterns float

ARGUMENTS

-rule_name *track_pattern_rule*

Specifies the track rule to use to create the tracks.

-layer *layer_name*

Specifies the layer to use for the tracks created by this track rule.

-start *point*

Specifies the location where the first track of the current pattern can start. If the tracks are vertical, the starting point specifies a x-coordinate. If the tracks are horizontal, the starting point specifies a y-coordinate.

The actual starting point depends on the locations of the previous SADP tracks on the given layer and the interactions between the different SADP tracks.

By default, the starting point is the left bottommost edge of the design.

-stop *point*

Specifies the location of the last track in the current pattern. If the tracks are vertical, the stopping point specifies an x-coordinate. If the tracks are horizontal, the stopping point specifies a y-coordinate.

The actual stopping point depends on the locations of the previous SADP tracks on the given layer and the interactions between different SADP tracks.

By default, the stopping point is the right topmost edge of the design.

-count ***number_of_repeats_using_rule***

Specifies the number of times the track rule is to be repeated.

-pitch ***pitch_between_patterns***

Specifies the spacing between the first track of adjacent track_patterns. The pitch should be used together with **count**.

-auto_fill intra_pattern | **inter_pattern** | **all** | **none**

Specifies how to fill spaces between routes that are large enough to accommodate one or more default tracks. Fill values are one of *intra_pattern*, *inter_pattern*, *all* or *none*.

Specify *intra_pattern* to create tracks between track groups or in between SADP tracks based on the same rule when there is sufficient space.

Specify *inter_pattern* to create tracks between two SADP tracks based on different SADP rules when there is sufficient space.

Specify *all* to fill all the intervening space with default width tracks.

Specify *none* to avoid filling intervening space.

By default, the auto fill value is *all*.

-exact

Start the SADP track exactly at the startpoint. You must specify **-start** together with this option. By default, the tool calculates the nearest allowed SADP location based on the track plan and other SADP tracks on the same layer.

-create_pg_route

Creates shapes for the entire length of track if the netname in the track pattern rule matches a PG netname.

-color mask_one | **mask_two**

Specifies the mask pattern for the track pattern.

DESCRIPTION

This command creates self-aligned double patterning (SADP) on the specified layer based on the track rule for the current block.

EXAMPLES

The following examples creates SADP tracks in the current block.

prompt> **generate_sadp_tracks -rule_name tr1 -start 40 -layer M7 -count 3**

{TRACK_0 TRACK_1 TRACK_2 TRACK_3 TRACK_4 TRACK_5}

SEE ALSO

check_sadp_tracks(2)
create_sadp_track_rule(2)
remove_sadp_track_rule(2)
report_sadp_track_rule(2)

generate_via_ladder_template

This command generates via ladder template according to input config file and current block.

SYNTAX

status **generate_via_ladder_template**
-config_file *name*
[-template_file *name*]
[-association_file *name*]

Data Types

name string

ARGUMENTS

-config_file *name*

Specifies the file-name of high-level input configuration in XML format. This option is required. An example of XML file is shown in EXAMPLES section.

-template_file *name*

Specifies the file-name of output via rule definition in TCL format. User can directly source the file to set via rule into data-base. The default file-name is auto_gen_via_ladder_template.tcl.

-association_file *name*

Specifies the file-name of output via ladder constraint association in TCL format. User can directly source the file to set the association into data-base. The default file-name is auto_gen_via_ladder_association.tcl.

DESCRIPTION

This command generates via ladder template according to input config file and current opened block. The config file should describe the structure of via ladder template in XML format. By taking design rules defined in the technology file into consideration, this command would generate corresponding TCL commands in side files, which could be sourced by user in later stage.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

```
prompt> generate_via_ladder_template \
   -config_file input.config \
   -template_file output_via_rule.tcl \
   -association_file output_via_association.tcl
```

Content of input.config :

```
<EmRule>
   <!—template definition -->
   <Template name="template_1_3231">
      <Layer name="M1" row_number="3"/>
      <Layer name="M2" row_number="2"/>
      <Layer name="M3" row_number="3"/>
      <Layer name="M4" row_number="1"/>
   </Template>

... more templates

   <!-- Pin to template association -->
   <Pin name="*/BCELLD5A11*/Z">
      <Template name="template_1_3231"/>
   </Pin>

.... more associations

</EMRule>
```

Content of output_via_rule.tcl :

```
create_via_rule -name template_1_3231 \
   -cut_layer_names {VIA1 VIA2 VIA3} -cut_names {Vs Vs Vs}
   -cut_rows {2 3 1} -cuts_per_row {3 2 3}
set viaRule [get_via_rules template_1_3231]
set_attribute $viaRule upper_metal_min_length_table {0.3 0.2 0.4}
set_attribute $viaRule cut_x_min_spacing_table {0.07 0.11 0.11}
set_attribute $viaRule cut_y_min_spacing_table {0.06 0.06 0.15}
set_attribute $viaRule max_num_stagger_tracks_table {1 2 0}
set_attribute $viaRule for_electro_migration true
```

output_via_association.tcl :

```
foreach_in_collection pin [get_lib_pins -quiet */BCELLD5A11*/Z] {
      set_attribute -quiet $pin is_em_via_ladder_required true
set_via_ladder_candidate $pin -ladder_name "template_1_3231"
}
```

SEE ALSO

generate_via_rules_for_performance

Generate via rules based on the analysis of routing layers.

SYNTAX

status **generate_via_rules_for_performance**
-max_layer layer_name pins
-rule_file rule_file_name
-stagger

Data Types

layer_name string
rule_file_name string

ARGUMENTS

-max_layes *layer_name*

Specifies max_layer of via ladders. This should not be higher layer than the max_layer in the routing rule.

-rule_file *rule_file_name*

Specifies output via rule file name. (Default: auto_perf_via_ladder_rule.tcl)

-stagger

Generate stagger table

DESCRIPTION

This command generates via rules based on the analysis of routing layers. This command returns 1 if succeeded, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the options.

EXAMPLES

The following is the usage examples.

```
prompt> generate_via_rules_for_performance
1

prompt> generate_via_rules_for_performance -stagger
1
```

SEE ALSO

create_via_rule(2)
remove_via_rules(2)
report_via_rules(2)
setup_performance_via_ladder(2)

get_abstract_type

Get the type of an abstract view design

SYNTAX

string **get_abstract_type**
[design]

ARGUMENTS

design

The abstract view design to get type from. Only abstract view design is accepted.

DESCRIPTION

Get the type of the specified abstract view design.

EXAMPLES

The following example gets the type of the abstract view of design BlkA:

prompt> **get_abstract_type [get_designs BlkA -filter "view_name == abstract"]**

SEE ALSO

create_abstract(2)

get_alternative_lib_cell

Creates a collection of equivalent library cells for the specified cell or library cell.

SYNTAX

collection **get_alternative_lib_cells**
[-current_library]
[-libraries *libraries*]
[-skip_pg]
[-skip_timing]
object

Data Types

libraries list
object list

ARGUMENTS

-current_library

If this option is specified, then searches for library cells in the cell's current library or library cell's current library only. You cannot specify this option with the -libraries option.

-libraries *libraries*

Specifies a list of libraries from which to select alternative library cells. The default is to select from all libraries in the link path. The libraries option can be a list of library names, or collections of libraries loaded; the latter can be obtained using the get_libs command. You cannot specify this option with the -current_library option.

-skip_pg

Skip pg pin consistency checking while searching the alternative cell. Then the return alternative lib cell pg pin setting could be different with the specified lib cell.

-skip_timing

Skip timing consistency checking while searching the alternative cell. Then the return alternative lib cell timing arcs could be different with the specified lib cell.

object

Specifies the cells or library cell for which to return equivalent library cells. This is a required option. Patterns is allowed, but the specified cell or library cell must be unique. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the wildcards man page.

DESCRIPTION

The **get_alternative_lib_cells** command creates a collection of library cells that are logically similar to the specified cell or library cell and pass the filter, if specified. This collection can be used to replace or resize the specified cell in the current design.

The command returns a collection if any library cells match the criteria. If no objects match the criteria, the command returns an empty string.

You can use the **get_alternative_lib_cells** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_alternative_lib_cells** result to a variable.

When issued from the command prompt, **get_alternative_lib_cells** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_alternative_lib_cells** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use the **get_alternative_lib_cells** command as an argument to the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the equivalent library cells for the slow/AND2X2 library cell

```
prompt> set libcelsel [get_lib_cells slow/AND2X2]
{slow/AND2X2}

prompt> [get_alternative_lib_cells $libcelsel]
{slow/AND2X1 slow/AND2X6 slow/AND2X12}
```

The following example queries the equivalent library cells for the top/U21 cell instance.

```
prompt> set celsel [get_cell top/U21]
{top/U21}

prompt> [get_alternative_lib_cells $celsel]
{slow/AND2X1 slow/AND2X2 slow/AND2X6 slow/AND2X12}
```

SEE ALSO

collections(2)
filter_collection(2)
get_cells(2)
get_lib_cells(2)
list_attributes(2)
query_objects(2)
shell.common.collection_result_display_limit(3)

get_alternative_lib_cells

Creates a collection of equivalent library cells for the specified cell or library cell.

SYNTAX

collection **get_alternative_lib_cells**
[-current_library]
[-libraries *libraries*]
[-skip_pg]
[-skip_timing]
object

Data Types

libraries list
object list

ARGUMENTS

-current_library

If this option is specified, then searches for library cells in the cell's current library or library cell's current library only. You cannot specify this option with the -libraries option.

-libraries *libraries*

Specifies a list of libraries from which to select alternative library cells. The default is to select from all libraries in the link path. The libraries option can be a list of library names, or collections of libraries loaded; the latter can be obtained using the get_libs command. You cannot specify this option with the -current_library option.

-skip_pg

Skip pg pin consistency checking while searching the alternative cell. Then the return alternative lib cell pg pin setting could be different with the specified lib cell.

-skip_timing

Skip timing consistency checking while searching the alternative cell. Then the return alternative lib cell timing arcs could be different with the specified lib cell.

object

Specifies the cells or library cell for which to return equivalent library cells. This is a required option. Patterns is allowed, but the specified cell or library cell must be unique. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the wildcards man page.

DESCRIPTION

The **get_alternative_lib_cells** command creates a collection of library cells that are logically similar to the specified cell or library cell and pass the filter, if specified. This collection can be used to replace or resize the specified cell in the current design.

The command returns a collection if any library cells match the criteria. If no objects match the criteria, the command returns an empty string.

You can use the **get_alternative_lib_cells** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_alternative_lib_cells** result to a variable.

When issued from the command prompt, **get_alternative_lib_cells** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_alternative_lib_cells** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use the **get_alternative_lib_cells** command as an argument to the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the equivalent library cells for the slow/AND2X2 library cell

```
prompt> set libcelsel [get_lib_cells slow/AND2X2]
{slow/AND2X2}

prompt> [get_alternative_lib_cells $libcelsel]
{slow/AND2X1 slow/AND2X6 slow/AND2X12}
```

The following example queries the equivalent library cells for the top/U21 cell instance.

```
prompt> set celsel [get_cell top/U21]
{top/U21}

prompt> [get_alternative_lib_cells $celsel]
{slow/AND2X1 slow/AND2X2 slow/AND2X6 slow/AND2X12}
```

SEE ALSO

collections(2)
filter_collection(2)
get_cells(2)
get_lib_cells(2)
list_attributes(2)
query_objects(2)
shell.common.collection_result_display_limit(3)

get_antenna_rule_names

Name

get_antenna_rule_names

Creates a list of antenna rule names defined in the library.

SYNTAX

list **get_antenna_rule_names**
[-mode *antenna_mode*]
[-library *library*]

Data Types

antenna_mode integer
library collection

ARGUMENTS

-mode *antenna_mode*

An integer value between 1 and 6, that represents the way the antenna areas are computed. When this option is specified, only the antenna rules names that use the given mode for antenna area calculation, are returned. This is an optional option. When not specified all the antenna rule names defined in the library are returned.

-library *library*

Specifies the library from which the antenna rule names are to be returned. This is an optional option. By default the command returns the antenna rule names defined in the current library.

DESCRIPTION

This command returns the antenna rule names for the specified mode that are defined in the given library.

EXAMPLES

The following example returns all the antenna rule names in the current library.

```
prompt> get_antenna_rule_names
{r1 r2 r3 r4 r5 r6}
```

The following example returns all the antenna rule names from the library lib1, that use mode 1 for antenna area calculation.

```
prompt> get_antenna_rule_names -mode 1 -library lib1
r1
```

SEE ALSO

define_antenna_rule(2)
define_antenna_layer_rule(2)
report_antenna_rules(2)
remove_antenna_rules(2)

get_app_option_value

Returns the value or the information of the specified application option.

SYNTAX

string **get_app_option_value**
[-block *block*]
[-user_default]
[-system_default]
[-details]
-name *name*

OLD SYNTAX

[-design *design*]
[-global]
[-default]
[-details]
-name *name*

Data Types

block block name or collection
name string

ARGUMENTS

-block *block*

Specifies the block for which the option values to be returned are set on. This option cannot be specified with the *-user_default* or the *-system_default* option. If this option is not specified, the application option value will be obtained from the current design, if any. If there is no current design or the specified option is not explicitly set on the design, then the option value will be obtained from the user-default or system-default (if the user-default is not set).

-user_default

Returns the user default value of the application option if set. If the user-default does not exist, then nothing is returned. This option is mutually exclusive with the *-system_default* option.

-system_default

Returns the system default value of the application option. This option is mutually exclusive with the *-user_default* option.

-details

Indicates that detailed information about the specified option is returned. If this option is specified, the command will return a list containing name and value pairs related to the option definition. This option cannot be specified with any of the *-user_default*, *-system_default* or *-block* options.

-name *name*

Specifies the name of the application option whose values is to be returned. For application options that are of type "list of {name value} pairs", this option also accepts the subscripts for pointed modifications. The subscripts should be encolsed within braces " ()" (e.g. optionName(subscript)). This is a mandatory option.

-design *design*

Specifies the design which the option values to be returned are set on. This option cannot be specified with the *-global* option. If neither *-design* nor *-global* is specified, option value will be obtained from the current design, if any. If there is no current design or the specified option is not explicitly set on the design, then the option value will be obtained from the global scope. This option is deprecated, use *-block* option instead.

-global

Indicates that the option values in the global scope will be returned. This option cannot be specified with the *-design* option. This option is deprecated.

-default

Indicates that the default value of the option is returned. This option cannot be specified with the *-details* option. This option will be obsolete, use *-user_default* or *-system_default* instead.

DESCRIPTION

This command returns the value of the specified application option. If the application option is design scoped, then the value for the block specified with *-block* is returned. With the *-details* option specified, this command returns the information of the option definition as a list of name and value pairs.

Each design will store only the options explicitly set on it. Any design scoped application option values not set on the design will be obtained from the user-default or system-default (if user-default is not set). When a default value is being returned, the value will first be validated for the block and if the value is valid, then it will be returned. If the value turns out to be not valid for the block, then an empty value is returned. For application options that are of type list, the default value returned for the block may have lesser values, since the invalid values are trimmed off.

For application options of type "list of {name value} pairs", values for keys can be selectively obtained by using the subscript mechanism.

For backward compatibility the older syntax of the command will still be supported, but the options are made hidden. The options '-design', '-global' and '-default' are made hidden.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the options.

EXAMPLES

The following example gets the value of the *time.enable_preset_clear_arcs* option on the current design (or in the global scope if there is no current design).

```
prompt> get_app_option_value -name time.enable_preset_clear_arcs
false
```

The following example gets the system-default value of the **shell.tmp_dir_path** application option.

```
prompt> get_app_option_value -system_default -name shell.tmp_dir_path
/tmp
```

The following example gets the user-default value of the **shell.tmp_dir_path** application option.

```
prompt> get_app_option_value -user_default -name shell.tmp_dir_path
/user/temp/
```

The following example gets the value for the M4 key of the **test.layer_name_number_pair_list** application option on the current block.

```
prompt> get_app_option_value -name test.layer_name_number_pair_list(M4)
34
```

The following example reports information about the **time.all_clocks_propagated** application option.

```
prompt> get_app_option_value -details -name time.all_clocks_propagated
name time.all_clocks_propagated value_type bool default_value false
help {Determines whether or not all clocks are created as propagated clocks}
status basic is_obsolete false is_persistent true
```

SEE ALSO

get_app_options(2)
report_app_options(2)
set_app_options(2)
reset_app_options(2)
help_app_options(2)

get_app_options

Returns a list of valid application option names.

SYNTAX

list **get_app_options**
[-block *block*]
[-global]
[-non_default]
[-quiet]
[*pattern*]

OLD SYNTAX

list **get_app_options**
[-design *design*]
[*pattern*]
[-quiet]

Data Types

block block name or collection
pattern string

ARGUMENTS

-block *block*

Specifies the block which the application options to be returned are set on. If this option is specified, the command returns only those options whose values are explicitly set on the specified block.

-global

Returns only the global scoped application options. This is an optional option. This option is mutually exclusive with the *-block* option. This option is deprecated.

-non_default

Returns app options set to different behavior than the system-default. This includes:

- Block-scope app options set to a non-default setting with set_app_options

- Global-scope app options set to a non-default setting with set_app_options
- Block-scope app options given a user default setting with set_app_options -as_user_default

-quiet

Suppresses warning or error messages.

pattern

Specifies a pattern for the option names. A pattern can include the wildcard characters asterisk (*) and question mark (?). "*" will match zero or more of any character, and "?" will match any single character. If this option is specified, the command returns those option names that match the specified pattern. If not specified, includes all options.

-design *design*

Specifies the design which the application options to be returned are set on. If this option is specified, the command returns only those options whose values are explicitly set on the specified design. This option is deprecated, use *-block* option instead.

DESCRIPTION

This command returns a list of application options whose names match the specified pattern. If a block is specified, only those options that are explicitly set on the block will be returned. The return value may be an empty list if there are no such options.

The default behavior is to return all the application options.

For backward compatibility the older syntax of the command will still be supported, but the options are made hidden. The option '-design', is made hidden.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets the list of all options that contain "dir" in their names.

```
prompt> get_app_options *dir*
shell.tmp_dir_path
```

The following example gets the list of all options in the *timer* category, which are explicitly set on the current block.

```
prompt> get_app_options timer.* -block [current_block]
time.all_clocks_propagated time.enable_preset_clear_arcs
```

The following example gets the list of all global scoped application options whose value is different from the default.

```
prompt> get_app_options -global -non_default
shell.man_path time.enable_preset_clear_arcs
```

SEE ALSO

get_app_option_value(2)
report_app_options(2)
set_app_options(2)
reset_app_options(2)
help_app_options(2)

get_app_var

Gets the value of an application variable.

SYNTAX

string **get_app_var**
[-default | -details | -list]
[-only_changed_vars]
var

Data Types

var string

ARGUMENTS

-default

Gets the default value.

-details

Gets additional variable information.

-list

Returns a list of variables matching the pattern. When this option is used, then the *var* argument is interpreted as a pattern instead of a variable name.

-only_changed_vars

Returns only the variables matching the pattern that are not set to their default values, when specified with **-list**.

var

Specifies the application variable to get.

DESCRIPTION

The **get_app_var** command returns the value of an application variable.

There are four legal forms for this command:

- get_app_var *<var>*
  Returns the current value of the variable.

- get_app_var *<var>* -default
  Returns the default value of the variable.

- get_app_var *<var>* -details

Returns more detailed information about the variable. See below for details.

- get_app_var -list [-only_changed_vars] *<pattern>*

Returns a list of variables matching the pattern. If *-only_changed_vars* is specified, then only variables that are changed from their default values are returned.

In all cases, if the specified variable is not an application variable, then a Tcl error is returned, unless the application variable **sh_allow_tcl_with_set_app_var** is set to true. See the **sh_allow_tcl_with_set_app_var** man page for details.

When **-details** is specified, the return value is a Tcl list that is suitable as input to the Tcl **array set** command. The returned value is a list with an even number of arguments. Each odd-numbered element in the list is a key, and each even-numbered element in the list is the value of the previous key.

The supported keys are as follows:

name

This key contains the name of the variable. This key is always present.

value

This key contains the current value of the variable. This key is always present.

default

This key contains the default value of the variable. This key is always present.

help

This key contains the help string for the variable. This key is always present, but sometimes the value is empty.

type

This key contains the type of the application variable. Legal values of for this key are: string, bool, int, real. This key is always present.

constraint

This key describes additional constraints placed on this variable. Legal values for this key are: none, list, range. This key is always present.

min

This key contains the min value of the application variable. This key is present if the constraint is range. The value of this key may be the empty string, in which case the variable only has a max value constraint.

max

This key contains the max value of the application variable. This key is present if the constraint is "range". The value of this key may be the empty string, in which case the variable only has a min value constraint.

list

This key contains the list of legal values for the application variable. This key is present if the constraint is "list".

EXAMPLES

The following are examples of the **get_app_var** command:

```
prompt> get_app_var sh_enable_page_mode
1

prompt> get_app_var sh_enable_page_mode -default
false

foreach {key val} [get_app_var sh_enable_page_mode -details] {   echo "$key: $val"
}
=>  name: sh_enable_page_mode
    value: 1
    default: false
    help: Displays long reports one page at a time
    type: bool
    constraint: none

prompt> get_app_var -list sh_*message
sh_new_variable_message
```

SEE ALSO

report_app_var(2)
set_app_var(2)
write_app_var(2)

get_attribute

Returns attribute values on the specified objects.

SYNTAX

```
list get_attribute
  [-class class_name]
  [-quiet]
  [-value_list]
  -objects object_list
  -name attribute_name
  pos_object_list
  pos_attribute_name

class_name          string
object_list         list
attribute_name      string
pos_object_list     list
pos_attribute_name  string
```

ARGUMENTS

-class *class_name*

Specifies the first class object type name used for implicit search of objects. This can be used when objects are specified as string and its object type is not among the predefined types in the search order set by app option *shell.common.implicit_find_mode*.

-quiet

Indicates that error and warning messages are not to be reported.

-value_list

Indicates that the result should always be returned as a list.

-objects *object_list*

Specifies a list of objects from which to get the attribute values. Each element in the list is a collection of objects. This option is mutually exclusive with *pos_object_list*. Either one (and only one) of *-objects* or *pos_object_list* must be specified.

-name *attribute_name*

Specifies the name of the attribute whose value is returned. This option is mutually exclusive with *pos_attribute_name*. Either one (and only one) of *-name* or *pos_attribute_name* must be specified.

pos_object_list

Specifies a list of objects from which to get the attribute values. Each element in the list is a collection of objects. This positional option is provided for compatibility with other tools.

pos_attribute_name

Specifies the name of the attribute whose value is returned. This positional option is provided for compatibility with other tools.

DESCRIPTION

This command searches the *object_list* for the specified attribute and returns a list of attribute values. An empty list is returned if the attribute is not found on any of the specified objects.

The *object_list* can be specified as strings in which case objects will be implicitly searched based on object type search order as specified by app option *shell.common.implicit_find_mode*. If none of the predefined object type matches specified input string then the command will fail. If the object type is known then user can use *-class* to provide that input.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets the value of the *rise_drive* attribute for the IN1 port.

```
prompt> get_attribute -objects [get_ports IN1] -name rise_drive
0.25
```

The above result can also be achieved by using implicit search with string input as shown below.

```
prompt> get_attribute -objects IN1 -name rise_drive
0.25
```

The following example gets the value of *is_primary* attribute for voltage area VA1 by specifying VA1 as string and using *-class* option since voltage area is not among predefined object types for implicit find search order.

```
prompt> get_attribute -objects VA1 -name is_primary -class voltage_area
true
```

The following example gets the value of the *load* attribute for all ports whose names begin with OUT:

```
prompt> get_attribute -objects [get_ports OUT*] -name load
1.0 2.0 2.5 1.0
```

The following example sets a user-defined attribute named X on some cells and then uses **get_attribute** commands to retrieve this attribute value, as well as the value of an application attribute.

```
prompt> define_user_attribute -type int -classes cell -name X
prompt> set_attributes -objects [get_cells *] -name X -value 30
{"i1", "i2"}
prompt> foreach_in_collection sel [get_cells *] {
?       echo -n "On '[get_attribute -objects $sel -name full_name]', "
?       echo "X =  [get_attribute -objects $sel -name X]"
```

```
?      }
On 'i1', X =  30
On 'i2', X =  30
```

The following example shows how the command works when some objects fail to get their attributes.

```
prompt> get_attribute -objects [get_cells {cell1 cell2 cell3}] -name area
Warning: Attribute 'area' does not exist on cell 'cell2' (ATTR-3)
79.833600 {} 79.833600
```

The following example shows the warning that is issued because the U9 cell is not found:

```
prompt> get_attribute -objects [get_cells {U1 U9}] -name dont_touch
Warning: No cell objects matched 'U9' (SEL-004)
true
```

SEE ALSO

define_user_attribute(2)
foreach_in_collection(2)
get_defined_attributes(2)
list_attributes(2)
remove_attributes(2)
report_attributes(2)
set_attribute(2)
shell.common.implicit_find_mode(3)

get_blackbox_generated_clocks

Returns a blackbox generated clock object that is created by the **create_blackbox_generated_clock** command.

SYNTAX

int **get_blackbox_generated_clocks**
[-filter *expression*]
[-quiet]
pattern

Data Types

expression string
pattern string

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any clocks that match *patterns* (or *objects*), the expression is evaluated based on the attributes of the clock. If the expression evaluates to true, the clock is included in the result.

-quiet

Suppresses warning and error messages if no objects match.

patterns

Matches clock names against patterns. Patterns can include the wildcard characters "*" and "?".

DESCRIPTION

This command returns blackbox generated clock objects as previously created by the **create_blackbox_generated_clock** command. Use this command in Tcl scripts to explore the characteristics of blackbox generated clock that you created. And you can use **get_attributes** to examine its attribute values.

EXAMPLES

The following example retrieves some attributes for the blackbox generated clock.

```
prompt> get_attribute -name master_clock [get_blackbox_generated_clocks GCK]
prompt> get_attribute -name master_source [get_blackbox_generated_clocks GCK]
prompt> get_attribute -name network_delay_max [get_blackbox_generated_clocks GCK]
prompt> get_attribute -name network_delay_min [get_blackbox_generated_clocks GCK]
prompt> get_attribute -name source_pin [get_blackbox_generated_clocks GCK]
prompt> get_attribute -name period [get_blackbox_generated_clocks GCK]
```

SEE ALSO

create_blackbox_generated_clock(2)

get_block_objects

Creates a collection of physical objects by making the input block as the top block.

SYNTAX

collection **get_block_objects**
[-filter *expression*]
[-quiet]
[-block *block_name*]
[*objects*]

Data Types

expression string
block_name string
objects list

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on physical object's attributes. You can determine the corresponding physical object's attributes by using the **list_attributes** command. If the expression evaluates to **true**, the object is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-block *block_name*

Specifies the block to use as the top block for returning the physical objects. The block should be in the instance tree of the physical objects been passed, otherwise it will be an error condition.

objects

Specifies name or collection of objects in the top design's context.

DESCRIPTION

This command creates a collection of physical objects by selecting objects from a homogeneous collection of objects after

converting it to input block's context and after that meet the selection criteria. It returns a collection handle if one or more physical objects meet the selection criteria. If no object match the selection criteria, it returns an empty string.

You can use the **get_block_objects** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable. See the **collections** command man page for information about working with collections.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets the shape in the input block's context.

```
prompt> set result [get_block_objects -block [get_blocks MID] TOP/MID/shape1]
prompt> get_attribute -objects $result -value full_name
{MID/shape1}
```

SEE ALSO

collections(2)
filter_collection(2)
get_blocks(2)
query_objects(2)
shell.common.collection_result_display_limit(3)

get_blocks

Creates a collection of one or more blocks in memory. You can assign these blocks to a variable or pass them into another command

SYNTAX

collection **get_blocks**
[-open | -explicit | -implicit | -all]
[-lib_cells]
[-hierarchical]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
block_patterns | -of_objects *objects*

Data Types

expression string
exact_count int
minimum_count int
block_patterns list
objects list

ARGUMENTS

-open

Returns only blocks that are currently open. This is the default.

-explicit

Returns only blocks that were explicitly opened with open_block.

-implicit

Returns only blocks that were implicitly opened.

-all

Includes unopened blocks, if matched.

-lib_cells

Includes lib_cell blocks, if matched.

-hierarchical

Searches only physical blocks in the current block hierarchy. Does not force an auto link.

-filter *expression*

Filters the collection with *expression*. For any modules that match *patterns*, the expression is evaluated based on the attributes for the design. If the expression evaluates to true, the design is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *objects*

Creates a collection of blocks associated with the specified objects. In this case, each object in the list is a name, pattern, or collection. The objects can be a lib, module or a design. **-of_objects** and *patterns* are mutually exclusive. You can specify only one of the two. If the specified object is a lib, blocks contained in the library are returned otherwise blocks which own the specified module or design is returned.

patterns

A collection of blocks, or a pattern to match block names against. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type block

DESCRIPTION

The **get_blocks** command creates a collection of blocks from those currently loaded into memory that match certain criteria. The command returns a collection if any blocks match the patterns and pass the filter (if specified). If no objects matched your criteria, the empty string is returned.

The pattern is of the form [[path_name/]lib_name:]block_name[/label_name][.view_name]. If the path_name is not given, all libraries that match the lib_name (if a lib_name specified), are searched. If the lib_name is not given, the current library is searched. If the view_name is not given, design views will be returned. If the label_name is not given, the default unnamed label will be returned. If the block_name is not given the pattern "*" is used.

Path_name can be absolute or relative. If a path name starts with a slash '/', it is interpreted as an absolute path name. If it starts with a tilde '~' or '~user', then it represents a user's home directory. If it starts with a '.', then it is relative to the current directory. Otherwise the path name is interpreted relative to the first component of the Tcl search_path variable which locates a file.

It is an error to use patterns and **-of_objects** option together.

The **-all** option allows get_blocks to return blocks that are not open. The **get_blocks** command will not open those blocks. Very little can be done with a block that is not open. An unopened block can be opened with **open_block**, or can be removed from its library with **remove_blocks**. Some attributes are available with **get_attribute**. An unopened block cannot be made the current block.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_blocks** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_blocks** result to a variable.

When issued from the command prompt, **get_blocks** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_blocks** provides a fast, simple way to display modules in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_blocks** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the modules that begin with 'mpu.'

```
prompt> get_blocks mpu*
{lib1:mpu_0_0.design lib1:mpu_0_1.design lib1:mpu_1_0.design}
```

The following example shows that, given a collection of blocks, you can remove those blocks.

```
prompt> remove_blocks [get_blocks mpu*]
```

```
Removing block lib1:mpu_0_0.design...
Removing block lib1:mpu_0_1.design...
Removing block lib1:mpu_1_0.design...
```

SEE ALSO

collections(2)
filter_collection(2)
current_block(2)
current_design(2)
open_block(2)
close_blocks(2)
get_designs(2)
remove_blocks(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_bound_shapes

Creates a collection by selecting bound shapes from the current design.

SYNTAX

collection **get_bound_shapes**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count int
minimum_count int
of_objects list
patterns string
region list
point list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with *expression*. For any bound shapes that match *patterns*, the expression is evaluated based on the bound shape's attributes. If the expression evaluates to true, the bound shape is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for bound shapes level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-of_objects *of_objects*

Creates a collection containing the bound shapes of the specified move bound. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

patterns

Matches the bound shape names in the current design against the specified patterns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-at *point*

Creates a collection that contains all bound shapes at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one

except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all bound shapes that are completely inside the specified region and doesn't include bound shapes which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all bound shapes that are completely inside the specified region and the bound shapes which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all bound shapes which are abut/touching from inside or outside to the specified region and the bound shapes whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of bound shapes of the specified move bounds. It returns a collection handle if one or more bound shapes are found. If no bound shapes are found, it returns an empty string.

You can use the **get_bound_shapes** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of bound shapes of the move bound "movebound1".

```
prompt> get_bound_shapes -of_objects "movebound1"
{"BOUND_SHAPE_1", "BOUND_SHAPE_2"}
```

SEE ALSO

add_to_bound(2)
collections(2)
create_bound(2)
create_bound_shape(2)
create_bound_shape(2)
filter_collection(2)
get_bound_shapes(2)
query_objects(2)
regexp(2)
remove_bound_shapes(2)
remove_bound_shapes(2)
remove_bounds(2)
remove_from_bound(2)
report_bounds(2)
shell.common.collection_result_display_limit(3)

get_bounds

Creates a collection by selecting bounds from the current design.

SYNTAX

collection **get_bounds**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count int
minimum_count int
of_objects list
patterns list
region list
point list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the bound attributes. You can determine the bound attributes by using the **list_attributes** command. If the expression evaluates to **true**, the bound is included

in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for bounds level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect**, and **-at**, but not with **-of_objects**.

-of_objects *of_objects*

Creates a collection containing the bounds of the specified cells, ports, or bound shapes. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

patterns

Matches the bound names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-at *point*

Creates a collection that contains all bounds at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all bounds that are completely inside the specified region and doesn't include bounds which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all bounds that are completely inside the specified region and the bounds which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all bounds which are abut/touching from inside or outside to the specified region and the bounds whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of bounds by selecting bounds that meet the selection criteria. It returns a collection handle if one or more bounds meet the selection criteria. If no bounds match the selection criteria, it returns an empty string.

You can use the **get_bounds** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of bounds by name pattern matching.

```
prompt> get_bounds movebound*
{"movebound1", "movebound2", "movebound3"}
```

The following example creates a collection of bound with attribute filtering.

```
prompt> get_bounds -filter {type == hard}
{"movebound1"}
```

The following example creates a collection of all bounds.

```
prompt> get_bounds *
{"movebound1", "movebound2", "movebound3", "groupboundA"}
```

The following example creates a collection of all bounds containing the cells "inv*" and port "reset".

```
prompt> get_bounds -of_objects "inv* reset"}
{"movebound1", "movebound2"}
```

SEE ALSO

add_to_bound(2)
collections(2)
create_bound(2)
create_bound_shape(2)
filter_collection(2)
get_bound_shapes(2)
query_objects(2)
regexp(2)
remove_bound_shapes(2)
remove_bounds(2)
remove_from_bound(2)

report_bounds(2)
shell.common.collection_result_display_limit(3)

get_budgets

Returns attributes, settings, and analysis for timing budgeting.

SYNTAX

int **get_budgets**
[-blocks]
[-clocks]
[-path_types]
[-all_segments]
[-pin]
[-all_pins]
[-pin_data]
[-all_pin_data]
[-pin_constraints]
[-all_pin_constraints]
[-fanin_segments *pin_data*]
[-fanout_segments *pin_data*]
[-fanin_cone *segments_or_pin_data*]
[-fanout_cone *segments_or_pin_data*]
[-of_pin *budgeted_block_pin*]
[-input]
[-output]
[-filter *expression*]

Data Types

pin_data budget_pin_data object
segments_or_pin_data collection
budgeted_block_pin string or list
expression string

ARGUMENTS

-blocks

Returns a Tcl collection of cells which have been selected for budgeting by the **set_budget_options -add_blocks** command.

-clocks

Returns a Tcl collection of budget_clock objects representing each budgeted clock in each budgeted block of the design. You can use collection and attribute commands to query data similar to that shown in the Clock summaries section of the **report_budget** report. See the budget_attributes man page for details on what data can be queried from a budget_clock_object. You can use the **-filter** to select certain objects, based on their attribute values.

-path_types

Returns a Tcl collection of budget_path_type objects representing each type of timing path in the budgeted design. You can use collection and attribute commands to query data similar to that shown by the **report_budget -path_type** command. See the budget_attributes man page for details on what data can be queried from a budget_path_type object. You can use the **-filter** to select certain objects, based on their attribute values.

-all_segments

Returns a Tcl list with the name of all segments in the budgeted design. A segment represents any subpath between two budgeted pins, or the worst timing path from/to a timing path startpoint/endpoint. You can use collection and attribute commands to query data similar to that shown by the **report_budget -fanin_segments** or **report_budget -fanout_segments** commands. See the budget_attributes man page for details on what data can be queried from a budget_segment object. You can use the **-filter** to select certain objects, based on their attribute values.

-pin

Returns a Tcl budget_pin object associated with the given budgeted pin of your design. A budget_pin object contains budget summary information for every pin on a budgeted block. You can use the budget_pin to query data similar to that shown by the **report_budget -pins** command. See the budget_attributes man page for details on what data can be queried from a budget_pin_data object. The **-pin** option must be accompanied by the by the **-of_pin** option.

-all_pins

Returns a collection of Tcl budget_pin objects that cover all pins of all budget blocks. These are the same type of objects return by the **-pin** option. You can use the **-filter** to select certain objects, based on their attribute values.

-pin_data

Returns a Tcl collection of budget_pin_data objects associated with the given budgeted pin of your design. A budget_pin_data object represents the derived budget for a single timing path through a budgeting pin. Note that one pin can have many timing paths passing through it, so this option might return many pin_data objects. You can use collection and attribute commands to query data similar to that shown by the **report_budget -pins** command. See the budget_attributes man page for details on what data can be queried from a budget_pin_data object. The **-pin_data** option must be accompanied by the **-of_pin** option.

-all_pin_data

Returns a collection of Tcl budget_pin_data objects that cover all data where a path passes through a budget pin. These are the same type of objects return by the **-pin_data** option. You can use the **-filter** to select certain objects, based on their attribute values.

-pin_constraints

Returns a Tcl collection of budget_pin_constraint objects associated with the given budgeted pin of your design. The command returns one budget_pin_constraint object for each unique constraint that has been specified with the **set_pin_budget_constraints** command. You can use collection and attribute commands to query data similar to that shown by the **report_budget -pins** command. See the budget_attributes man page for details on what data can be queried from a budget_pin_constraint object. The **-pin_constraints** option must be accompanied by the **-of_pin** option.

-all_pin_constraints

Returns a collection of Tcl budget_constraint objects that cover all constraints applied to budgeted pins. These are the same type of objects return by the **-pin_constraints** option. You can use the **-filter** option to select certain objects, based on their attribute values.

-fanin_segments ***pin_data***

Returns a collection of budget_segment objects that "fan into" the given budget_pin_data object. You can use this option to trace from segments at the end of a budget path to segments nearer to the beginning of a budget path. You can use collection and attribute commands to query data similar to that shown by the **report_budget -fanin_segments** or **report_budget -**

fanout_segments commands. See the budget_attributes man page for details on what data can be queried from a budget_segment object.

-fanout_segments ***pin_data***

Returns a collection of budget_segment objects that "fan out of" the given budget_pin_data object. You can use this option to trace from segments at the beginning of a budget path toward segments nearer to the end of a budget path. You can use collection and attribute commands to query data similar to that shown by the **report_budget -fanin_segments** or **report_budget -fanout_segments** commands. See the budget_attributes man page for details on what data can be queried from a budget_segment object.

-fanin_cone ***segments_or_pin_data***

Returns a collection of budget_segment objects that occur in the transitive fanin of the given budget_segment or budget_pin_data objects. If you give segments to this option, those segments will be included in the resulting collection. The fanin_cone option can be combined with the fanout_cone option to get all budget_segments between two sets of budget_segment or budget_pin_data objects.

-fanout_cone ***segments_or_pin_data***

Returns a collection of budget_segment objects that occur in the transitive fanout of the given budget_segment or budget_pin_data objects. If you give segments to this option, those segments will be included in the resulting collection. The fanout_cone option can be combined with the fanin_cone option to get all budget_segments between two sets of budget_segment or budget_pin_data objects.

-of_pin ***budgeted_block_pin***

Specify a pin or a list of pins to be used with the **-pin**, **-pin_data** or **-pin_constraints** option. This option can be specified with the **-input** or **-output** options.

-input

Returns data for the "input" direction of a bidirectional pin. Specify this option with the **-of_pin** option.

-output

Returns data for the "output" direction of a bidirectional pin. Specify this option with the **-of_pin** option.

-filter ***expression***

Filters the collection with *expression*. Only objects that conform to the expression will be included in the output collection. The **-filter** option can be used with the **-clocks**, **-path_types**, **-all_segments**, **-all_pins**, **-all_pin_data**, or **-all_pin_constraints** option.

DESCRIPTION

Queries attributes, settings, and analysis information for budgeting. Almost any data that can be obtained from the **report_budget** command can be accessed as Tcl objects and attributes by the **get_budgets** command. See the different option descriptions for details.

Use commands such as **foreach_in_collection**, **index_collection**, and **get_attribute** to access the raw data. See the budget_attributes man page for details of what attributes are available on what objects.

EXAMPLES

The following example prints a list of segments that have frozen constraints.

```
set fpins [get_budgets -all_pins -filter internal_frozen==true]
foreach_in_collection pin $fpins {
   echo [get_attribute $pin pin.full_name]
}
```

The following example prints a list of pins on budgeted blocks that have no timing paths.

```
set upins [get_budgets -all_pins -filter undefined(pin_data)]
foreach_in_collection pin $upins {
   echo [get_attribute $pin pin.full_name]
}
```

The following example prints a list of budget path types that are launched from CLK2.

```
set pts [get_budgets -path_types -filter launch_clock.clock.name==CLK2]
foreach_in_collection pt $pts {
   echo [get_attribute $pt full_name]
}
```

The following example prints a list of budget segments that are constrained by the "fast_rule" net_estimation_rule.

```
set pts [get_budgets -all_segments -filter rule_name==fast_rule]
foreach_in_collection pt $pts {
   echo [get_attribute $pt full_name]
}
```

The following example prints the number of budgeted paths in the design that start in "block1" and end in "block2".

```
set count 0
foreach_in_collection pin [get_pins block1/*] {
 foreach_in_collection pin_data [get_budgets -pin_data -of_pin $pin] {
  foreach_in_collection segment [get_budgets -fanout_segments $pin_data] {
   set to_cell [get_cells -of_objects [get_attribute $segment to_pin]]
   if {[get_attribute $to_cell name] == "block2"} {
    set count [expr $count + 1]
   }
  }
 }
}
puts $count
```

The following example prints the name of any budgeted pin where a multicycle path through the pin is not meeting the current budget.

```
foreach_in_collection pin_data [get_budgets -all_pin_data] {
 set path_type [get_attribute $pin_data path_type]
 set is_multicycle [string equal -length 19 \
  [get_attribute $path_type exception] "set_multicycle_path"]
 if {$is_multicycle && (
    ([get_attribute $pin_data launch_slack] < 0.0) ||
    ([get_attribute $pin_data capture_slack] < 0.0))} {
  puts $pin
 }
}
```

The following example returns all segments on paths between block1 and block2.

```
set from_segs [get_budget -all_segments -filter \
  "from_block==block1 AND undefined(from_pin)"]
set to_segs [get_budget -all_segments -filter \
  "to_block==block2 AND undefined(to_pin)"]
set between_segs [get_budgets -fanout_cone $from_segs -fanin_cone $to_segs]
```

SEE ALSO

budget_attributes(3)
collections(2)
compute_budget_constraints(2)
get_attribute(2)
report_attributes(2)
report_budget(2)
set_boundary_budget_constraints(2)
set_budget_options(2)
set_latency_budget_constraints(2)
set_pin_budget_constraints(2)
write_budgets(2)

get_bump_cluster_name

Returns a string of one or more bump cluster names which match the specified pattern, or which contain any of the specified *-of_objects*. If more than one bump cluster match input criteria, the return string contains space-separated name of all bump clusters.

SYNTAX

string **get_bump_cluster_name**
[*patterns*
| -of_objects *objects*]

Data Types

patterns string or collection
objects collection

ARGUMENTS

patterns

Specifies the name pattern for finding bump clusters. This is an optional argument. If a wildcard expression is specified, it is used to match against all bump clusters in current design.

-of_objects *objects*

Specifies a collection of one or more objects for which a bump cluster containing such object is matched. Allowed objects are only those objects which can be contained in an edit group. This is an optional argument. This argument is mutually exclusive with *patterns* argument. If no argument is specified, command returns the name of all bump clusters.

DESCRIPTION

This command returns a space-separated string containing the names of all bump clusters which match the input criteria. Bump clusters can be matched either by their name for given wildcard expression, or by the *-of_objects* option. If no bump cluster matches the input criteria, the command issues a warning message and returns an empty string.

EXAMPLES

The following example returns a space-separated string of all bump clusters with a name that begins with bump_cluster_1:

```
prompt> get_bump_cluster_name bump_cluster_1*
"bump_cluster_1 bump_cluster_10 bump_cluster_11 bump_cluster_12"
```

The following examples returns a space-separated string of all bump clusters with name that contain specified **-of_objects**:

```
prompt> get_bump_cluster_name -of_objects [get_cells bump_cluster_xyz_bump23]
"bump_cluster_xyz"
```

SEE ALSO

create_bump_cluster(2)
get_bump_cluster_objects(2)

get_bump_cluster_objects

Creates a collection by selecting specified type of objects from given bump cluster(s).

SYNTAX

collection **get_bump_cluster_objects**
cluster_name_or_collection
[*-bumps*
| *-tsvs*
| *-vias*
| *-route_shapes*]

Data Types

cluster_name_or_collection string or collection

ARGUMENTS

cluster_name_or_collection

Specifies the bump cluster name or collection for finding objects. This is a required argument. If a wildcard expression is specified, it is used to match against all bump clusters in the current design.

-bumps

Returns all bump cell instances for given bump cluster collection. This is an optional argument. However, if none of the other optional arguments is specified, command selects this option as default behavior.

-tsvs

Returns all TSV cell instances for a given bump cluster collection. This is an optional argument.

-vias

Returns all vias for a given bump cluster collection. Note, vias do not include TSV instances. This is an optional argument.

-route_shapes

Returns all route shapes for a given bump cluster collection. This is an optional argument.

DESCRIPTION

This command creates a collection of the specified type of objects from the bump clusters that match the bump cluster name, or are defined by the bump cluster collection. It returns a collection handle if one or more specified type of objects are found. If no objects are found, it issues a warning message and returns an empty string.

Use the **get_bump_cluster_objects** command as an argument to another command or assign its result to a variable. See the example below for more information.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of all non-TSV cells contained in bump cluster bump_cluster_123:

```
prompt> get_bump_cluster_objects bump_cluster_123
{"bump_cluster_123_bump1", "bump_cluster_123_bump2", "bump_cluster_123_bump3",
"bump_cluster_123_bump4"}
```

The following examples creates a collection of all TSV cells contained in bump clusters whose name begins with bump_cluster_xyz:

```
prompt> get_bump_cluster_objects bump_cluster_xyz* -tsvs
{"THROUGH_SILICON_VIA_0", "THROUGH_SILICON_VIA_1"}
```

The following examples creates a collection of all non-TSV vias contained in all bump clusters:

```
prompt> get_bump_cluster_objects * -vias
{"VIA_S_0", "VIA_S_1"}
```

The following examples creates a collection of all route shapes contained in bump clusters of given collection:

```
prompt> get_bump_cluster_objects [get_edit_groups bump_cluster_xyz] -route_shapes
{"RECT_74_1", "RECT_74_2"}
```

SEE ALSO

create_bump_cluster(2)
get_bump_cluster_name(2)
get_edit_groups(2)

get_bundles

Creates a collection of bundles. You can assign these bundles to a variable or pass them into another command.

SYNTAX

collection **get_bundles**
[-design *design*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-filter *expression*]
patterns | -of_objects *objects*

Data Types

design collection
exact_count int
minimum_count int
expression string
patterns list
objects list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-filter *expression*

Filters the collection with *expression*. For any bundles that match *patterns* (or *objects*) the expression is evaluated based on the bundle's attributes. If the expression evaluates to true, the bundle is included in the result.

patterns

Matches bundle names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type bundle.

-of_objects *objects*

Creates a collection of bundles containing the specified objects. Each object is a named bundle or net, or bundle collection or net collection. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_bundles** command creates a collection of bundles in the current design, that match certain criteria. The command returns a collection if any bundles match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

If any *patterns* fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_bundles** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_bundles** result to a variable.

When issued from the command prompt, **get_bundles** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable **collection_result_display_limit**.

The "implicit query" property of **get_bundles** provides a fast, simple way to display bundles in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_bundles** as an

argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the bundles that begin with "o" and reference an FD2 library bundle. Although the output looks like a list, it is not. The output is just a display.

```
prompt> get_bundles "BUNDLE*"
{"BUNDLE_0", "BUNDLE_3"}
```

SEE ALSO

add_to_bundle(2)
collections(2)
create_bundle(2)
filter_collection(2)
remove_bundles(2)
remove_from_bundle(2)
report_bundles(2)

get_busplans

Get busplan_bus objects as defined by the **create_busplans** command.

SYNTAX

int **get_busplans**
[-of_object pin_port]
[pattern] [-filter *expression*]
[-quiet]

Data Types

pattern string
expression string

ARGUMENTS

-of_object *pin_or_port*

Returns the busplan that passes through the given pin or port, if any.

patterns

Matches bus names against patterns. Patterns can include the wildcard characters "*" and "?". If no pattern is specified, it defaults to "*". Patterns can also include collections of type busplan_bus. *patterns* and **-of_objects** are mutually exclusive; you must specify one, but not both.

-filter *expression*

Filters the collection with *expression*. For any buses that match *patterns* (or *objects*), the expression is evaluated based on the attributes of the bus. If the expression evaluates to true, the bus is included in the result.

-quiet

Suppresses warning and error messages if no matching busplans are found.

DESCRIPTION

Return busplan_bus objects as previously defined by the **create_busplans** command.

EXAMPLES

Get the busplan named bus1.

prompt> **get_busplans bus1**

Get all busplans with a rule named "fast_rule".

prompt> **get_busplans * -filter "rule_name==fast_rule"**

SEE ALSO

set_net_estimation_rule(2)
create_busplans(2)

get_cell

Creates a collection of cells from the current design relative to the current instance. You can assign these cells to a variable or pass them into another command.

SYNTAX

collection **get_cells**
[-design *design*]
[-hierarchical]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-physical_context]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-filter *expression*]
[-hsc *separator*]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
exact_count int
minimum_count int
expression string
patterns list
separator string
objects list
region list
point list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-hierarchical

Searches for cells level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX **find** command. For example, if there is a cell block1/adder, a hierarchical search finds it using "adder". This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-physical_context

Specifies that only the physical cells be searched and returned. This option can also be used with **-of_objects**. If *objects* argument to **-of_objects** is a pin, than the cell owning the pin is returned. If *objects* argument to **-of_objects** is a net, than a collection of cells owning the physical pins connected to the net is returned. If *objects* argument to **-of_objects** is a physical hierarchical cell, a collection of physical cells contained in the hierarchical cell is returned. If **-of_objects**, is not used than physical cells matching the specified *patterns* are returned.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-filter *expression*

Filters the collection with *expression*. For any cells that match *patterns* (or *objects*) the expression is evaluated based on the cell's attributes. If the expression evaluates to true, the cell is included in the result.

-hsc *separator*

Specifies a separator character. The default is /. Valid values are /, @, ^, #, ., and |.

-of_objects *objects*

Creates a collection of cells connected or associated to the specified objects. In this case, each object in the list is a name, pattern, or collection. The objects can be a cell, pin, net, bound, io_guide, io_ring, lib_cell, module, voltage_area, voltage_area_shape, edit_group, rp_group, power_strategy, placement attraction, stub_chains, scan_chains, safety register rule,

or safety register group.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

In addition, **-hierarchical** with **-of_objects**, only makes sense if the objects are cells or rp_groups.

patterns

Matches cell names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type cell.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at ***point***

Creates a collection that contains all cells at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within ***region***

Creates a collection that contains all cells that are completely inside the specified region and doesn't include cells which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching ***region***

Creates a collection that contains all cells that are completely inside the specified region and the cells which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one

except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect ***region***

Creates a collection that contains all cells which are abut/touching from inside or outside to the specified region and the cells whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

The **get_cells** command creates a collection of cells in the current design, relative to the current instance, that match certain criteria. The command returns a collection if any cells match the *patterns* or *objects* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

If any *patterns* (or *objects*) fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_cells** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_cells** result to a variable.

When issued from the command prompt, **get_cells** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable **collection_result_display_limit**.

The "implicit query" property of **get_cells** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_cells** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the cells that begin with "o" and reference an FD2 library cell. Although the output looks like a list, it is not. The output is just a display.

```
prompt> get_cells "o*" -filter "ref_name == FD2"
{"o_reg1", "o_reg2", "o_reg3", "o_reg4"}
```

The following example shows that, given a collection of pins, you can query the cells connected to those pins.

```
prompt> set pinsel [get_pins o*/CP]
{"o_reg1/CP", "o_reg2/CP"}
prompt> query_objects [get_cells -of_objects $pinsel]
{"o_reg1", "o_reg2"}
```

The following example sets a timing derate on cells i1 and i2.

```
prompt> set_timing_derate -late 1.1 [get_cells {i1 i2}]
1
```

SEE ALSO

collections(2)
filter_collection(2)
get_pins(2)
link_block(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_cell_array_patterns

Creates a collection by selecting cell array_patterns from the current block.

SYNTAX

collection **get_cell_array_patterns**
[-filter *expression*]
[-quiet]
[-design *design_name*]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]

Data Types

expression string
design_name string
exact_count integer
minimum_count integer
patterns list

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*, which must be a boolean expression based on the cell array pattern attributes. You can determine the cell array pattern attributes by using the **list_attributes** command. If the expression evaluates to **true**, the cell array pattern is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-design *design_name*

Specifies the design module in which to search for cell array patterns.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are

mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for cell array patterns level-by-level relative to the current instance.

patterns

Matches the cell array pattern names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

DESCRIPTION

This command creates a collection of cell array patterns that meet the selection criteria. It returns a collection handle if one or more cell array patterns meet the selection criteria. If no cell array patterns match the selection criteria, it returns an empty string.

You can use the **get_cell_array_patterns** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of cell_array_patterns by matching the name against the pattern:

```
prompt> get_cell_array_patterns CELL_ARRAY_PATTERN*
{CELL_ARRAY_PATTERN0 CELL_ARRAY_PATTERN1 CELL_ARRAY_PATTERN2}
```

The following example creates a collection of all cell_array_patterns:

```
prompt> get_cell_array_patterns *
{mycap0 mycap1 CELL_ARRAY_PATTERN0 CELL_ARRAY_PATTERN1 CELL_ARRAY_PATTERN2}.in -0.25in
```

SEE ALSO

create_cell_array_pattern(2)
remove_cell_array_patterns(2)
report_cell_array_patterns(2)

get_cell_buses

Creates a collection of cell_buses from the current design relative to the current instance. You can assign these cell_buses to a variable or pass them into another command.

SYNTAX

collection **get_cell_buses**
[-design *design*]
[-hierarchical]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-filter *expression*]
[-hsc *separator*]
patterns | -of_objects *objects*

Data Types

design collection
exact_count int
minimum_count int
expression string
patterns list
separator string
objects list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-hierarchical

Searches for cell_buses level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX **find** command.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-filter *expression*

Filters the collection with *expression*. For any cell_buses that match *patterns* (or *objects*) the expression is evaluated based on the cell_bus's attributes. If the expression evaluates to true, the cell_bus is included in the result.

-hsc *separator*

Specifies a separator character. The default is /. Valid values are /, @, ^, #, ., and |.

-of_objects *objects*

Creates a collection of cell_buses associated to the specified objects. In this case, each object in the list is a name, pattern, or collection. The objects can be a cell or pin_bus. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both. In addition, **-hierarchical** with **-of_objects**, only makes sense if the objects are cells or rp_groups.

patterns

Matches cell_bus names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type cell_bus. Supports patterns containing hierarchical separator when **-hierarchical** is specified.*patterns* and **-of_objects** are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_cell_buses** command creates a collection of cell_buses in the current design, relative to the current instance, that match certain criteria. The command returns a collection if any cell_buses match the *patterns* or *objects* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

A cell_bus represents an array of cells created with the same base name. This is typically accomplished in the source HDL using an

array notation.

If any *patterns* (or *objects*) fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_cell_buses** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_cell_buses** result to a variable.

When issued from the command prompt, **get_cell_buses** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable **collection_result_display_limit**.

The "implicit query" property of **get_cell_buses** provides a fast, simple way to display cell_buses in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_cell_buses** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the cell_buses that begin with "reg_bus". Although the output looks like a list, it is not. The output is just a display.

```
prompt> get_cell_buses "reg_bus*"
{"reg_bus_x", "reg_bus_y"}
```

SEE ALSO

collections(2)
create_cell_bus(2)
filter_collection(2)
get_cells(2)
link_block(2)
query_objects(2)
regexp(2)
remove_cell_buses(2)
shell.common.collection_result_display_limit(3)

get_cells

Creates a collection of cells from the current design relative to the current instance. You can assign these cells to a variable or pass them into another command.

SYNTAX

collection **get_cells**
[-design *design*]
[-hierarchical]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-physical_context]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-filter *expression*]
[-hsc *separator*]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
exact_count int
minimum_count int
expression string
patterns list
separator string
objects list
region list
point list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-hierarchical

Searches for cells level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX **find** command. For example, if there is a cell block1/adder, a hierarchical search finds it using "adder". This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-physical_context

Specifies that only the physical cells be searched and returned. This option can also be used with **-of_objects**. If *objects* argument to **-of_objects** is a pin, than the cell owning the pin is returned. If *objects* argument to **-of_objects** is a net, than a collection of cells owning the physical pins connected to the net is returned. If *objects* argument to **-of_objects** is a physical hierarchical cell, a collection of physical cells contained in the hierarchical cell is returned. If **-of_objects**, is not used than physical cells matching the specified *patterns* are returned.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-filter *expression*

Filters the collection with *expression*. For any cells that match *patterns* (or *objects*) the expression is evaluated based on the cell's attributes. If the expression evaluates to true, the cell is included in the result.

-hsc *separator*

Specifies a separator character. The default is /. Valid values are /, @, ^, #, ., and |.

-of_objects *objects*

Creates a collection of cells connected or associated to the specified objects. In this case, each object in the list is a name, pattern, or collection. The objects can be a cell, pin, net, bound, io_guide, io_ring, lib_cell, module, voltage_area, voltage_area_shape, edit_group, rp_group, power_strategy, placement attraction, stub_chains, scan_chains, safety register rule,

or safety register group.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

In addition, **-hierarchical** with **-of_objects**, only makes sense if the objects are cells or rp_groups.

patterns

Matches cell names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type cell.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at ***point***

Creates a collection that contains all cells at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within ***region***

Creates a collection that contains all cells that are completely inside the specified region and doesn't include cells which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching ***region***

Creates a collection that contains all cells that are completely inside the specified region and the cells which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one

except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all cells which are abut/touching from inside or outside to the specified region and the cells whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

The **get_cells** command creates a collection of cells in the current design, relative to the current instance, that match certain criteria. The command returns a collection if any cells match the *patterns* or *objects* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

If any *patterns* (or *objects*) fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_cells** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_cells** result to a variable.

When issued from the command prompt, **get_cells** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable **collection_result_display_limit**.

The "implicit query" property of **get_cells** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_cells** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the cells that begin with "o" and reference an FD2 library cell. Although the output looks like a list, it is not. The output is just a display.

```
prompt> get_cells "o*" -filter "ref_name == FD2"
{"o_reg1", "o_reg2", "o_reg3", "o_reg4"}
```

The following example shows that, given a collection of pins, you can query the cells connected to those pins.

```
prompt> set pinsel [get_pins o*/CP]
{"o_reg1/CP", "o_reg2/CP"}
prompt> query_objects [get_cells -of_objects $pinsel]
{"o_reg1", "o_reg2"}
```

The following example sets a timing derate on cells i1 and i2.

```
prompt> set_timing_derate -late 1.1 [get_cells {i1 i2}]
1
```

SEE ALSO

collections(2)
filter_collection(2)
get_pins(2)
link_block(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_cells_of_scan_chain

Returns a collection containing all scan cells of the specified SCANDEF stub chain.

SYNTAX

collection **get_cells_of_scan_chain**
-chain *chain_name*

Data Types

chain_name string

ARGUMENTS

-chain *chain_name*

Specifies the name of the SCANDEF stub chain.

DESCRIPTION

This command returns a Tcl collection containing all scan cells for the specified scan chain.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example gets all scan cells of the *chain1* scan chain:

prompt> **get_cells_of_scan_chain -chain "chain1"**

SEE ALSO

get_scan_chain_count(2)

get_checkpoint_data

Queries the checkpoints and the associated actions and reports.

SYNTAX

string **get_checkpoint_data**
[-list_names]
[-name *checkpoint_name*]
[-list_reports]
[-report *report_name*]
[-list_actions]
[-action *action_name*]

Data Types

checkpoint_name string
report_name string
action_name string

ARGUMENTS

-list_names

Specifies a time-ordered list of previously executed checkpoints.

-name *checkpoint_name*

Specifies detailed information about a previously executed checkpoint.

-list_reports

Lists all defined checkpoint system reports.

-report *report_name*

Specifies detailed information about a defined checkpoint report.

-list_actions

Lists all defined checkpoint system actions.

-action *action_name*

Returns detailed information about a defined checkpoint action.

DESCRIPTION

This command queries information about the checkpoint system, including previously executed checkpoints, defined reports and actions, as well as details about those checkpoints, reports, and actions.

Use -list_names, -list_reports, or -list_actions to get a Tcl list of all executed checkpoints or defined reports or actions. Use -name, -report, or -action to get detailed information about a specific checkpoint, report, or action. Specify only one of those six options at a time; all are mutually exclusive with each other.

The -name, -report, and -action options all return a Tcl dictionary consisting of key / value pairs of a particular property of that checkpoint / report / action, and the value for that property.

- memory: The peak memory up through execution of the checkpoint
- start_time: The starting clock time of that checkpoint's execution
- end_time: The ending clock time of that checkpoint's execution
- before_runtime: The runtime from the end of the previous checkpoint to the start of this checkpoint. For the first checkpoint in the tool session, it is the runtime from tool startup until the start of this checkpoint.
- before_report_runtime: The runtime of all before phase reporting for this checkpoint
- self_runtime: The runtime of the checkpointed block of Tcl commands, as well as any before/after/replace actions associated to that checkpoint
- after_report_runtime: The runtime of all after phase reporting for this checkpoint
- after_runtime: The runtime from the end of this checkpoint until the start of the next checkpoint. For the last checkpoint in an active tool session, it will always show zero. For the last checkpoint in an exited tool session, it will show the runtime from teh end of this checkpoint until the end of the tool session.
- before_reports: A time-ordered Tcl list of all reports that were executed before this checkpoint
- after_reports: A time-ordered Tcl list of all reports that were executed after this checkpoint
- before_actions: A time-ordered Tcl list of all actions that were executed before this checkpoint
- after_actions: A time-ordered Tcl list of all actions that were executed after this checkpoint

  after_patterns: A ordered Tcl list of association constraints to run before this report or action
- replace_patterns: A ordered Tcl list of association constraints to run before this action. -report does not return the replace_patterns property, since reports cannot be associated to replace a checkpoint.

  Each of the three *_patterns properties returns with this format:

  {{enable|disable1 pattern1} {enable|disable2 pattern2} ...}

  For example, this indicates the report or action was enabled for all checkpoints (*), and then disabled only for the create_placement checkpoint. {{enable *} {disable create_placement}}

EXAMPLES

The following example returns a time-ordered list of previously executed checkpoints.

```
prompt> get_checkpoint_data -list_names
```

The follow example returns information about the multibit report

```
prompt> get_checkpoint_data -report multibit
```

SEE ALSO

create_checkpoint_action(2)
remove_checkpoint_actions(2)
create_checkpoint_report(2)
remove_checkpoint_reports(2)
associate_checkpoint_action(2)
associate_checkpoint_report(2)
set_checkpoint_options(2)
eval_checkpoint(2)
reset_checkpoints(2)
get_current_checkpoint(2)

get_clock_balance_groups

Create a collect of clock_balance_groups in the current design.

SYNTAX

collection **get_clock_balance_groups**
[-design *design*]
[-mode *mode*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *objects*]

Data Types

design collection
mode collection
expression string
exact_count integer
minimum_count integer
objects collection
patterns list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-mode *mode*

Specifies the mode for finding objects. If this is not specified, objects will be found in the current mode.

-filter *expression*

Filters the collection with *expression*. For any clock_balance_group that matches *patterns* (or *objects*) the expression is evaluated based on the clock_balance_group's attributes. If the expression evaluates to true, the clock_balance_group is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *objects*

Creates a collection of clock_balance_groups associated with the specified objects. The object can be a clock. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

patterns

Matches clock_balance_group names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type clock_balance_group. *patterns* and **-of_objects** are mutually exclusive.

DESCRIPTION

This command will create a collection of clock balance groups in the current design and current mode.

EXAMPLES

The following example gets all clock_balance_groups in the mode "mode1".

prompt> **get_clock_balance_groups -mode mode1**

SEE ALSO

create_clock_balance_group(2)
report_clock_balance_groups(2)
remove_clock_balance_groups(2)
balance_clock_groups(2)

get_clock_gate_pins

Creates a collection of clock gate pins of the specified type.

SYNTAX

collection **get_clock_gate_pins**
[-type *pin_type*]
[-of_objects *clock_gates*]

Data Types

pin_type list
clock_gates list

ARGUMENTS

-type *pin_type*

Specifies pin type of the clock gating cells to be returned. The values can be:

gated_clock gated clock output pin
clock clock input pin
enable enable input pin
test test input pin
observation observation output pin

If you do not specify this option, the command returns gated_clock output pins.

-of_objects *clock_gates*

Specifies the clock gating cells whose pins are returned. If you do not specify this option, the command searches all clock gates.

DESCRIPTION

The **get_clock_gate_pins** command creates a collection of specified pins of clock gate cells in the current design. If no clock gate cells with the specified pin are found, an empty collection is returned.

You can use the **get_clock_gate_pins** command at the command prompt, or you can nest it as an argument to another command, such as **connect_net**. In addition, you can assign the **get_clock_gate_pins** result to a variable.

When issued from the command prompt, **get_clock_gate_pins** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum by using the

collection_result_display_limit variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The first example gets the gated clock output pin (default pin type) of the clockGateA cell.

prompt> **get_clock_gate_pins -of_objects {clockGateA}**

{clockGateA/ECK}

The next example returns the clock input pin and the enable pin of clockGateA and clockGateB.

prompt> **get_clock_gate_pins -type {clock enable}**

{clockGateA/CK clockGateA/E clockGateB/CK clockGateB/E}

SEE ALSO

collections(2)
get_clock_gates(2)
report_clock_gating(2)

get_clock_gates

Creates a collection of clock gate cells that meet the specified criteria.

SYNTAX

collection **get_clock_gates**
[-clock *clocks*]
[-level *levels*]
[-stage *stages*]
[-type {*clock_gate* | *self_gate*}]
[-origin {*tool_inserted* | *pre_existing*}]
[*patterns*]

Data Types

clocks list
levels list
patterns list

ARGUMENTS

-clock *clocks*

Searches for clock gate cells whose base clock matches *clocks*. If you do not specify this option, all clocks are searched.

-level *levels*

Searches for clock gates cells whose level matches the entry of *levels*. If you do not specify this option, all levels are searched.

-stage *stages*

Searches for clock gates cells whose stage matches the entry of *stages*. If you do not specify this option, all stages are searched.

-type {*clock_gate* | *self_gate*}

Specifies the type of gate cells to be obtained. The type can be *clock_gate*, *self_gate*, or both. With *clock_gate* the command searches for regular clock gate cells. With *self_gate* the command searches for self gate cells. If *clock_gate* and *self_gate* are used together, or neither are used, it searches for regular clock gates and self gates.

-origin {*tool_inserted* | *pre_existing*}

Specifies the origin of the clock gates to be obtained. The origin can be *tool_inserted*, *pre_existing*, or both. With *tool_inserted* the command searches for clock gate cells that are inserted by the tool. With *pre_existing* the command searches for clock gate cells that are manually inserted by the user. If *tool_inserted* and *pre_existing* are used together, or neither are used, it searches for pre-existing and tool inserted clock gates.

patterns

Matches clock gate names against *patterns*. Patterns can include the wildcard characters asterisk(*) and question mark(?). For more details about using and escaping wildcards, refer to the **wildcards** man page. If you do not specify any pattern, the tool uses (*).

DESCRIPTION

The **get_clock_gates** command creates a collection of clock gate cells in the current design that match the specified criteria. If no clock gate cells match the criteria, an empty collection is returned.

You can use the **get_clock_gates** command at the command prompt, or you can nest it as an argument to another command, such as **get_clock_gate_pins**. In addition, you can assign the **get_clock_gates** result to a variable.

When issued from the command prompt, **get_clock_gates** behaves as though **query_objects** is called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum by using the **shell.common.collection_result_display_limit** application option.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns clock gate cells in the design with the following configuration:

- Two clock gate cells manually inserted at level 1, clockGateA and clockGateB.
- Two clock gate cells inserted by the tool at level 2, clock_gate_mid_A/out_reg_0 and clock_gate_mid_B/out_reg_1.
- clockGateA is driving clock_gate_midA/out_reg_0.
- clockGateB is driving clock_gate_midB/out_reg_1.
- The clock source of clockGateA is CLKA.
- The clock source of clockGateB is CLKB.

```
prompt> get_clock_gates
{clockGateA clockGateB clock_gate_mid_A/out_reg_0 clock_gate_mid_B/out_reg_1}

prompt> get_clock_gates clock_gate_mid_A/*
{clock_gate_mid_A/out_reg_0}

prompt> get_clock_gates -level 1
{clockGateA clockGateB}

prompt> get_clock_gates -origin {tool_inserted}
{clock_gate_mid_A/out_reg_0 clock_gate_mid_B/out_reg_1}

prompt> get_clock_gates -clock CLKA
{clockGateA clock_gate_mid_A/out_reg_0}
```

SEE ALSO

collections(2)
report_clock_gating(2)
get_clock_gate_pins(2)
wildcards(3)

get_clock_group_groups

Creates a collection of the clock group groups (groups of a clock group) in a clock_group in the current scenario.

SYNTAX

collection **get_clock_group_groups** [-quiet]
clock_group

string *clock_group*

ARGUMENTS

clock_group

The clock group object for which to return the clock group group objects. Only one clock group object is allowed in this command.

DESCRIPTION

The **get_clock_group_groups** command creates a collection of clock group group objects of a specific clock group object in the current scenario. You can assign these clock group groups to a variable or pass them into another command. The clock groups were previously created by the **set_clock_groups** command.

The command returns a collection if the specified clock group object exists, else returns an empty string.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command works on the mode of the specified clock group.

EXAMPLES

In the following example a variable named cg is set to a clock group object. It then sets a variable named cg_groups with the clock group group objects of that clock group.

```
prompt> set cg_groups [get_clock_group_groups $cg]
```

SEE ALSO

collections(2)
set_clock_groups(2)
shell.common.collection_result_display_limit(3)

get_clock_groups

Creates a collection of clock groups in the current scenario.

SYNTAX

collection **get_clock_groups** [-quiet]
[-design *design*]
[-mode *mode*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *of_objects*]

Data Types

design collection
mode collection
expression string
exact_count int
minimum_count int
of_objects list
patterns list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-mode *mode*

Specifies the mode for finding objects. If this is not specified, objects will be found in the current mode.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-of_objects *of_objects*

Creates a collection of clock groups that contain the specified objects. Each object is a named clock or clock collection. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

-filter *expression*

Filters the collection with *expression*. For any clock group that match the pattern or the of_objects criteria, the expression is evaluated based on the clock group's attributes. If the expression evaluates to true, the clock group is included in the result.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches clock group names against patterns. Patterns can include the wildcard characters "*" and "?". *patterns* and **-of_objects** are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_clock_groups** command creates a collection of clock groups in the current scenario that match certain criteria. You can assign these clock groups to a variable or pass them into another command. The clock groups are created with the **set_clock_group** command. by default, a clock is not in a clock group.

The command returns a collection if any clock group object matches the pattern or the **-of_objects** specifiers and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

You can use the **get_clock_groups** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_clock_groups** result to a variable.

When issued from the command prompt, the **get_clock_groups** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this

maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_clock_groups** provides a fast, simple way to display clock groups in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_clock_groups** as an argument to **query_objects**. Note that the name shown for a clock group is either the one given during this clock group's specification and if not then its generated by the tool.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-mode** option.

EXAMPLES

The following example sets the variable cg to a collection of asynchronous clock groups which contains the clock clk1.

```
prompt> create_clock_group -asynchronous -group {clk1 clk2 clk3} -group {clk1a clk2a, clk3a}
prompt> set cg [get_clock_groups -of_objects {clk1} -filter "clock_group_type==asynchronous"]
```

SEE ALSO

collections(2)
set_clock_groups(2)
shell.common.collection_result_display_limit(3)

get_clock_skew_groups

Creates a collection of skew groups for a design. You can assign these skew groups to a variable or pass them into another command.

SYNTAX

collection **get_clock_skew_groups**
[-design *design*]
[-mode *mode_list*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-filter *expression*]
patterns | -of_objects *objects*

Data Types

design string
mode_list list
exact_count integer
minimum_count integer
expression string
patterns list
objects collection

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-mode *mode_list*

Specifies the mode for finding objects. If this is not specified, objects will be found in the current mode.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, it modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

When combined with the *-regexp* option, it makes matches case-insensitive.

-filter *expression*

Filters the collection with the *expression* value. For any skew groups that match the *patterns* values, the expression is evaluated based on the skew group's attributes. If the expression evaluates to true, the skew group is included in the result.

patterns

Matches skew_group names against patterns. Patterns can include the wildcard characters "*" and "?". *patterns* and **-of_objects** are mutually exclusive; you must specify one, but not both.

-of_objects *objects*

Creates a collection of skew_groups created on the specified objects. Objects can be pin and/or port and/or clock. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_clock_skew_groups** command creates a collection of skew groups that match certain criteria. The command returns a collection if any skew groups match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

You can use the **get_clock_skew_groups** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_clock_skew_groups** result to a variable.

When issued from the command prompt, the **get_clock_skew_groups** command behaves as though the **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of the **get_clock_skew_groups** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_clock_skew_groups** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-mode** option.

EXAMPLES

The following returns all the skew groups that are on the clock clk1.

```
prompt> get_clock_skew_groups -of_objects clk1
{"SG_clk11", "SG_clk2"}
```

SEE ALSO

collections(2)
create_clock_skew_group(2)
foreach_in_collection(2)
query_objects(2)
report_clock_skew_groups(2)
shell.common.collection_result_display_limit(3)

get_clock_tree_pins

Returns a collection of pins from one or more clock trees according to the specified criteria.

SYNTAX

collection **get_clock_tree_pins**
[-clocks *clock_list*]
[-scenario *scenario_list*]
[-mode *mode_list*]
[-corner *corner_list*]

[-from *from_list*]
[-through *through_list*]
[-to *to_list*]

[-groups_from *from_list*]

[-of_objects *pin_specifying_list*]

[-filter *extended_attribute_expression*]

[-sort_by *extended_attribute_expression*]

[-index_range {*start_index* [*end_index*]]

[-value_for_undefined_attributes *float*]

[-metrics *comma_separated_expression_list*]

[-assign_to_variable *string*]

[-statistics_variable *string*]

[-total *string*]

[-unique_cells]

[-unique_nets]

[-scan_all_hierarchical_pins]

[-expect *int* | -expect_at_least *int*]
[-verbose]
[-quiet]

Data Types

extended_attribute_expression string
comma_separated_expression_list list
pin_specifying_list collection
start_index integer
end_index integer
range_list list
from_list list
through_list list
to_list list

ARGUMENTS

-clocks *clock_list*

Reports the clock trees specified in *clock_list*. By default, the command reports for clocks in the current scenario. Note that a clock object returned by "get_clocks" is bound to a scenario, so if "get_clocks" command is used, a clock from a different scenario may be searched. If a textual clock name is supplied as an option, it is equivalent to using "get_clocks" on that name with no options. This option may not be used with the scenarios, modes or corners options.

-scenario *scenario_list*

Specifies the scenarios used for resolving which clock and mode to be employed.

-mode *mode_list*

Do not process any scenario that does not incorporate a mode from this list.

-corner *corner_list*

Do not process any scenario that does not incorporate a corner from this list.

-from *from_list*

Only clock elements that are driven, directly or indirectly, by these pins will be considered. Using -from will impact the way that the clock trees are traversed. Please see discussion below. Hierarchical pins are valid inputs to **-from** option.

-through *through_list*

Only clock elements on paths through these pins will be considered.

-to *to_list*

Only paths that go to these elements these pins will be considered.

-groups_from *from_list*

Return a list of collections, one collection for each pin listed here.

-of_objects *pin_specifying_list*

Returns only the pins connected to the specified pins, ports, cells, and nets.

-filter *extended_attribute_expression*

Specifies the expression with which to filter the results. The expression must use the standard pin attributes or the ephemeral pin attributes provided by this command.

The command evaluates the expression for each pin, and eliminates any pin where the value is zero or negative. This is similar

to the **-filter** option of the **get_pins** command, but additional ephemeral attributes are available, as described below.

If the expression has an unrecogized attribute name in it, the command prints a list of available attributes and error and return nothing.

-sort_by ***extended_attribute_expression***

Specifies the expression with which to sort the results. The expression must use the standard pin attributes or the ephemeral pin attributes provided by this command.

The command evaluates the expression for each pin, and sorts the pins based on its value.

-index_range { start end }

Allows you to access a subset of the pins. The starting index is 0 and the index of the end member is the size of the set minus 1. If the start value is negative, the range is interpreted as being relative to the end of the set.

-value_for_undefined_attributes ***float***

By default, if the filter or sorting operation accesses an attribute that is undefined on a pin, such as the height of a pin that has no downstream loads, that pin will be excluded from the active set of pins. This can be overridden by the "value_for_undefined_attributes" parameter, in which case the attribute is replaced by the specified value. If this is not specified, undefined attributes have a value of -999.0. Note that an undefined attribute is different than an unknown attribute.

-metrics ***comma_separated_expression_list***

The -metrics option allows you to gather many attributes at once. The results are formed into a Tcl list and stored in the Tcl array variable specified with the -assign option. Note that if -assign is used without -metrics, the expression specified in -sort_by is stored. Any number of expressions may be specified in the -metrics option, such as "is_net_driver", "depth_max-depth_min", "toggle_rate*capacitance_max". Most commonly, the attributes are simply listed, without arithmetic operators, and calculations are done after the command is finished. The -metrics option must be used with the -assign_to_variable option.

-assign_to_variable ***tcl_variable_name***

By specifying a Tcl variable name with the assign_to_variable option, the tool will create an associative array variable by that name. This is a way to capture the value of attribute expressions. The keys of this array will be the full name of each pin returned. If used with -sort_by, this stores the floating point value of the attribute expression used to sort the pins. If used with -metrics, any number of attribute expressions may be specified. The resulting floating point values will be stored as a Tcl list in each element of the array.

-statistics_variable ***tcl_variable_name***

The -sort_by and -metrics options compute the floating point value of expressions for each clock pin output. Some statistics computed over these values, such as their max, min and mean, will be captured as a Tcl list in the variable specified by the statistics_variable option.

-total ***string***

This must be used with a -sort_by option. It computes the total of all sort_by expressions and assigns the result into the specified Tcl variable. Note that the command will also print this total, along with the smallest, largest, mean, and standard drivation of these values, whenever the command is run with a -sort_by expression.

-unique_cells

If the tool encounters multiple pins on the same cell more than once in the filtered output list, only the first occurance will be considered, and subsequent pins on the same cell will be discarded. This is useful for getting totals of attribute values, since totaling the attribute values of both inputs and outputs leads to double counting.

-unique_nets

If the tool encounters multiple pins on the same net more than once in the filtered output list, only the first occurance will be

considered, and subsequent pins on the same net will be discarded. This is useful for getting totals of attribute values, since totaling the attribute values of both the inputs and outputs of a cell causes double counting.

-scan_all_hierarchical_pins

By default, the tool skips over hierarchical pins unless they have a clock source defined on them. If -scan_all_hierarchical_pins is specified, the tool scans them, and if they are not filtered out, they are returned. Note that the value of some attributes, such as latency_max, are not defined on hierarchical pins.

-expect *exact_count*

Expect exactly this many matching objects, and raise a Tcl exception if this is not found. (Value of exact_count must be >= 0)

-expect_at_least *minimum_count*

Expect at least this many matching objects, and raise a Tcl exception if this is not found. (Value of minimum_count must be >= 0)

-verbose

Print more information about the pins found. If used with the "-sort_by" option, print a report of all the pins found and the value of the "sort_by" expression.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

DESCRIPTION

The **get_clock_tree_pins** command returns selected pins from the clock trees in the design. By default, all pins on all clock trees will be returned. The options allow you to restrict the set to a smaller set of pins. It can be used before or after CTS.

The command operates in up to seven phases:

1) Clock tree traversal, and construction of the working set of pins

2) Removing any pins from the working set if they not on paths specified by the from, through or to options.

3) Removing any pins from the working set if they were not included in the -of_objects option.

4) Removing any pins from the working set if the "filter" option expression cannot be evaluated or evaluates to false.

5) Sorting the working set according to the attribute expression specified.

6) Selecting a sub-set of the sorted pins by index.

7) Returning the working set as a collection of pins, and optionally by Tcl variable.

The -clocks, -scenario, -from, and -mode options control the way that the tool traverses the clock tree(s). In a large design with many clocks, using these options may speed up execution. As it traverses the tree(s) it builds up a set of pins internally. The other options control which of these pins will be returned.

The -from, -through, and -to options can be used to discard any pins that are not on paths that are from, or though, or to specified pins. In order to qualify, a path must meet all the from, through, to requirements simultaneously. That is, if a pin is on the "from" list in one clock and on the "to" list in another clock, those pins would not quality and would be filtered out.

The -groups_from option returns not just one collection of pins, but a list of collections. The tool works through each specified pin,

and finds the clock pins that are driven from it. This is typically used to find the sinks driven by each of a list of driver pins. Each of these collections is appended into a list, which is returned by the command. If the fanouts of the pins listed overlap each other, the downstream driver is "preferred", so sinks, for example, will appear associated with the pin that drives them most closely. This is mutually exclusive with the -from, -through, and -to options.

The "-of_objects" option allows you to restrict the set of clock pins to pins related to the specified objects. The objects may be cells, nets, ports or other pins. This makes it possible, for example, to identify which pins on a given cell are "active" in the specified clock tree(s). This may not be obvious if the cell is a macro or a large multi-input gate. Note that you can pass the output of one invocation of get_clock_tree_pins to be the input to a second invocation by using this option.

The -filter option allows you to deselect pins based on the standard attributes of pins (as is done with the get_pins commands), and/or a set of ephemeral attributes available only in this command. The ephemeral attributes provide access to clock specific information, so, for example, you can filter out pins based on their latency.

The -sort_by option controls the way the set of pins is sorted. Once sorted, the pins are ordered and you can select a sub-range of them.

Please note that you can assemble groups of pins using get_clock_tree_pins several times and joining the results with "add_to_collection -unique".

Attributes available only in this command

Many of the normal pin attributes such as is_net_driver and is_net_load are available for use with the -filter option. These are listed in the next section. This section describes some additional ephemeral attributes that are available inside get_clock_tree_pins command only: they can be used to control inclusion or exclusion from the results, and can be made available for design analysis.

The ephemeral attributes include:

capacitance_limit transition_limit

These are float attributes that may be defined on all pins. CTS has several ways of specifying max capacitance, max transition rules based on clock and designs. These attributes return the value that is in effect for CTS under the specified set of clocks and scenarios. If there is no rule, the attribute is undefined.

capacitance_max

This is an ephemeral float attribute that may not be defined on all pins. The attribute represents a simple, lumped model of the pin's total load capacitance, including the net and the load pins. If a pin is inside a physical macro or for some other reason has no meaningful capacitance, this value will be undefined.

ccd_offset

This attribute represents the offset set by ccd for sinks as part of balance point. For sinks with no -offset component, it reports a "0" value. For non-sinks and intermediate pins on clock network, it report a garbage value, -999

cell_num_pins

The number of pins on the cell that the pin is attached to.

cell_internal_power_max cell_leakage_power_max cell_switching_power_max cell_total_power_max

These are ephemeral float attributes that may not be defined on all pins. Since power is associated with cells, not pins, these attributes refer to the power disappated by the cell that the pin belongs to. The total power is the sum of leakage, internal and switching power. Power is the highest power rate observed on the cell across all scenarios accessed by the command. (Note that the leakage_power, internal_power, switching_power and total_power attributes are also available at the shell level with get_attribute. These access the power in the current scenario.) Note that if a cell is accessed from more than one pin, the power attribute will be accessed twice. If you are trying to compute the total power of cells, use the "-unique_cells" option with these

attributes.

center_x center_y

These are ephemeral distances (i.e. float) attributes that are defined on all physical pins. They are the XY coordinates for the center of the pin. It may be useful for filtering out pins that are not in the region of interest, or for sorting the results from left to right, for example.

depth_max depth_min height_max height_min

These are ephemeral integer attributes that may not be defined on all pins. The depth of pin is the number of nets between it and the clock tree source. All the pins on a (single driver) net have the same depth_max and depth_min, and the pins on the net attached to the root have a depth_max and depth_min of 0. Each pin as a max and min depth because there may be reconvergence in the tree (but on most pins the depth_max will equal the depth_min). If there is no path from a clock root to the pin, the depth attributes are undefined.

The height of pin is the number of nets between it and a clock balanced pin (sink) below it in the tree. In general, the pins on a net will have different heights, and the pins attached to a sink have a height_min and height_max of 0. If there is no path from a pin to a timing balanced pin, the height attributes are undefined.

downstream_delay_max downstream_delay_min

These are ephemeral float attributes that may not be defined on all pins. The downstream delay (aka "phase delay") is computed from a pin to all of the time balanced sink pins that it drives, and includes any float pin timing. It does not include any time to "drc only" pins downstream. If no sinks are found downstream of a pin, the downstream delays are undefined.

The rise/fall times cannot be accessed independently with these attributes.

fanout_max

This is an ephemeral integer attribute that is defined on output pins. The fanout of a pin is the number of loads it drives. The fanout_max is the max fanout over all paths that pass though the output pin.

is_beyond_clock_boundary

This is an ephemeral integer (0 or 1) attribute defined on all pins. This is 1 for pins encountered while traversing the clock tree after a clock balance point has been encountered.

is_cell_sizable

This is an ephemeral integer (0 or 1) attribute defined on all pins. This has the value 1 if the cell that the pin is attached to can be resized by CTS.

is_clock_source

This is an ephemeral integer (0 or 1) attribute defined on all pins. If a generated clock lists this pin as its source, the is_generated_clock_source attribute will be 1. Note that if the command is run with several scenarios/modes and a pin is considered a clock source in any mode, then this attribute will be 1.

is_clock_to_data

This is an ephemeral integer (0 or 1) attribute defined on all pins. If a clock path transitions from being a clock element to data elements, the pin where this happens will have the clock_to_data attribute equal to 1.

is_connected_to_multidriven_net

This is an ephemeral integer (0 or 1) attribute defined on all pins. This is 1 for all driver and load pins connected to a multi-driven

net (including disabled mesh drivers). Given a pin, if its net has multiple drivers, then it will be filtered by this attribute.

is_cts_added

This an ephemeral integer (0 or 1) attribute defined on all pins. This attribute is 1 for pins on buffers and inverters that were added by clock tree synthesis.

is_float_pin is_explicit_ignore

These are ephemeral integer (0 or 1) attributes defined on all pins. A float pin may have a balance_point stop on it with an explicit time, or without a time (aka a stop pin). A pin may be explicitly ignored, or it may be ignored because of traversal rules.

is_generated_clock_source

This is an ephemeral integer (0 or 1) attribute defined on all pins. If a generated clock lists this pin as its source, the is_generated_clock_source will be 1. Note that if the command is run with several scenarios/modes and a pin is considered a generated clock source in any mode, then this attribute will be 1.

is_global_routed

This is a integer (0 or 1) attribute defined on all pins. After global routing, most nets will have some global routing associated with them. But other nets (such as very short ones), may not. Also, depending on the flow, your script may have routed a subset of clock nets, or the design might have been modified after global routing was run. The is_globally_routed property can filter nets based on this.

is_ignored

This is an ephemeral integer (0 or 1) attribute defined on all pins. Ignored pins do not participate in CTS.

is_inside_etm

This is an ephemeral integer (0 or 1) attribute defined on all pins. It is 1 for pins found inside an ETM object.

is_internal

This is an ephemeral integer (0 or 1) attribute defined on all pins. An internal pin is neither an net_driver nor a net_load.

is_leaf is_sink

These are ephemeral integer (0 or 1) attributes defined on all pins. All the pins that transition as a result of a root pin transition, and which do not drive any further pins are leaf pins. All pins that have a controlled timing relationship between them and a root are sinks. For a simple tree with no exceptions, these attributes are the same. Balance points are sinks. Note that if the command is run with several scenarios/modes and a pin is considered a sink in any mode, then this attribute will be 1.

is_mscts_subtree_driver_object

This is an ephemeral integer (0 or 1) attribute defined on all pins. This is 1 for all pins/ports specified with set_multisource_clock_subtree_options -driver_objects or set_multisource_clock_tap_options -driver_objects.

is_on_buffer is_on_icg is_on_inverter is_on_repeater

These are ephemeral integer (0 or 1) attributes defined on all pins. If a pin is on either a buffer or an inverter it is considered to be a on a repeater. Or a pin may be on an ICG, or on a gate other than an ICG (such as a NAND gate). For ports of the current design none of these attributes would be 1.

is_on_global_clock_tree

This is an ephemeral integer (0 or 1) attribute defined on all pins. This is 1 for all pins whose cell is inserted by create_clock_drivers or synthesize_multisource_global_clock_trees command

is_on_macro

This an ephemeral integer (0 or 1) attribute defined on all pins. If a pin is on a macro (physical block) this attribute will be 1.

is_on_physical_boundary

This is an ephemeral integer (0 or 1) attribute defined on all pins. If a pin is an hierarchical pin and resides on a physical boundry, this attribute will be 1. Note that this attribute should be used with the -scan_all_hierarchical_pins.

is_reconvergent

This is an ephemeral integer (0 or 1) attribute defined on all pins. A point driven by two clocks is called a convergent pin, and a point driven by two paths from the same clock is called a reconvergent pin. This attribute is 1 at the point of reconvergence and at pins below it in the tree.

is_removable

This is an integer (0 or 1) attribute defined on all pins. Pins on dont_touch cells are not removable.

is_root

This is an integer (0 or 1) attribute defined on all pins. This is 1 for all the pins where traversal starts for this command. Generally this is clock root pins, but if the -from option is used, the pins in the -from option have a is_root attribute value of 1.

latency_max latency_min

These are ephemeral float attributes that may not be defined on all pins. The max delay from a transition to a pin is available with the latency_max attribute and the latency_min is similarly defined. The latency of the root is the clock latency, which may be zero. If there is no path from the root, the value is undefined. After synthesis, latency is generally defined on all pins, even those that are not balanced by CTS such as DRC only pins past exceptions. Before synthesis, the latency value may be large, random or undefined (because of huge fanouts or massive DRC violations, for example.)

The rise/fall times cannot be accessed independently with these attributes.

load_pin_capacitance_max

This is an ephemeral float attribute that is defined on input pins. This is the max load capacitance of the pin.

net_num_pins

This is an ephemeral integer attribute defined on all pins. This is the number of pins on the net that the pin belongs to.

net_switching_power_max

This is an ephemeral float attribute that may not be defined on all pins. The power of a net is the charging/discharging power associated with the net. The power is the highest power rate observed on the net across all scenarios accessed by the command. Note that if a net is accessed from more than one pin, the power attribute will be reported twice. If you are trying to compute the total power of nets, use the "-unique_nets" option to avoid double counting.

num_edges_from_root_max

This is an ephemeral integer attribute defined on all pins. As a tree is traversed from the root, each timing arc is counted. The value is stored in this attribute. Sorting on this attribute and using the -to and -verbose options results in a list something like "report_clock_timing".

num_icgs_from_root_max

This is an ephemeral integer attribute defined on all pins. As a tree is traversed from the root, each time an ICG is encountered this value is incremented.

num_nonrepeaters_from_root_max

This is an ephemeral integer attribute defined on all pins. As a tree is traversed from the root, each time an ICG, MUX, or other clock gate is encountered this value is incremented.

routing_skew_max

This is an ephemeral float attribute that is defined on output pins. If the pin has only one fanout, its routing_skew is 0. If it has multiple fanouts, the routing_skew is the difference between the earliest and latest arrival times (which must be due to routing delays).

skew_max skew_min

This is an ephemeral float attribute that may not be defined on all pins. The skew of a pin is equal to the downstream delay max - downstream delay min that is observed as the CTS engine scans the tree. If the pin is encounted several times (e.g. is is part of several scenarios or clocks) then the skew_max may not equal the downstream_delay_max attribute - downstream_delay_min attribute. If there is one sink below the pin, the skew is 0.0. If there is no sink with a valid latency, the skew is undefined. Before synthesis, the skew attribute is generally random or undefined.

subtree_num_repeaters_max subtree_total_area_max

These are ephemeral attributes defined on the root pin of each subtree traversed. A subtree begins at the output of a non repeater (e.g. an ICG) and ends at a sink, a generated_clock, or the input of a non-repeater. The subtree_num_repeaters metric returns the number of repeater cells (inverters + buffers) in the subtree. If the subtree consists of an ICG that drives sinks directly this number will be zero. The subtree_total_area_max attribute returns the area of all repeaters plus the area of the root cell.

subtree_total_leakage_power_max subtree_total_internal_power_max subtree_total_switching_power_max

These are ephemeral float attributes defined on the root pins of subtrees. A subtree begins at the output of a non repeater (e.g. an ICG) and ends at a sink, a generated_clock, or the input of a non-repeater. These metrics return the total leakage, internal and switching power of the subtree, including the clock cell at its root. If the tool scans the driver pin and its subtree more than once and finds multiple values for the power, the largest value is returned.

toggle_rate_max

This is an ephemeral float attribute that may not be defined on all pins. It is equal to the highest switching toggle rate observed on the pin across all scenarios accessed by the command.

total_load_pin_capacitance_max

This is an ephemeral float attribute defined on net driver pins. This is the sum of the load pin capacitances.

toggle_rate_reduction_min

This is an ephemeral float attribute defined in the output of ICGs. This is the toggle rate of the output of an ICG divided by the the toggle rate of the input. If the ICG is not enabled, this will be zero. If it is enabled all the time, this rate will be 1.0. Most ICGs will have values between 0 and 1. If the value is greater than 1, there is probably an error in the SAIF file.

transition_max transition_min

These are ephemeral float attributes that may not be defined on all pins. These attributes represent the pin's max and min transition times, considering rise and fall, and early and late transitions.. The rise/fall times cannot be accessed independently with these attributes.

undefined

This is defined on all pins for convenience. When the tool must return a value for an undefined attribute, it returns -999 (or whatever value has been set with the -value_for_undefined_attributes option). Using "==undefined" in your filter express to handle undefined attributes (instead of "==-999") makes it a little easier to read. For example "-filter is_hierarchical==undefined" will return all pins for which the is_hierarchical attribute is not defined: e.g. ports.

Access to Persistent (non ephemeral) Attributes

In addition to the attributes listed above, the command provides access to some of the standard pin attributes. This can be useful when sorting or filtering pins. (The "is_net_driver" and "is_net_load" attributes are particularly useful.):

case_value clock_latency_fall_max clock_latency_fall_min clock_latency_rise_max clock_latency_rise_min clock_source_latency_early_fall_max clock_source_latency_early_fall_min clock_source_latency_early_rise_max clock_source_latency_early_rise_min clock_source_latency_late_fall_max clock_source_latency_late_fall_min clock_source_latency_late_rise_max clock_source_latency_late_rise_min constant_value dont_touch_network dont_touch_network_no_propagate hold_uncertainty is_async_pin is_clear_pin is_clock_gating_clock is_clock_gating_enable is_clock_gating_pin is_clock_pin is_clock_used_as_clock is_clock_used_as_data is_data_pin is_fall_edge_triggered_clock_pin is_fall_edge_triggered_data_pin is_feedthrough_port is_fixed is_hierarchical is_ideal is_latch_loop_breaker is_negative_level_sensitive_clock_pin is_negative_level_sensitive_data_pin is_net_driver is_net_load is_positive_level_sensitive_clock_pin is_positive_level_sensitive_data_pin is_preset_pin is_rise_edge_triggered_clock_pin is_rise_edge_triggered_data_pin is_scan is_three_state is_three_state_enable_pin is_three_state_output_pin is_user_latch_loop_breaker max_capacitance_constraint max_rise_input_cap max_rise_load_cap max_slack max_time_borrow max_transition_constraint min_capacitance_constraint min_slack propagated_clock setup_uncertainty user_case_value user_case_value user_clock_sense

Sorting by an Attribute Expression

The default behavior is to sort the pins in increasing alphabetical order before returning them. In addition, they may also be sorted by an expression using any boolean, integer or float attribute. For example, sorting on latency_max will yield a collection with the lowest latency pins first. For attributes such as is_sink and is_on_repeater which would normally be thought of as booleans, "false" has a value of 0, and is therefore smaller than "true" which has a value of 1, so pins with the false attribute value will appear in the output before pins with a "true" value. In addition to 0 and 1 values, some attributes may be undefined on some pins and when this occurs, they will have the value of -999 (or whatever value was set with the "-value_for_undefined_attributes" option). Pins with the same attribute value will be sorted alphabetically (i.e. "ba3/reg7/clk" before "g1/clk"). Note that you can reverse the sort by putting a minus sign in front of the expression. For example, sorting on -latency_max will yield a collection with the highest latency pins first. Pins with the same latency value will still be sorted forward alphabetically.

Selecting a subset of pins by index

If you don't want the whole set, you can specify which members you want by index. The starting index is 0, and if there are 7 members, the index of the last pin is 6. The range {0 9} will try to return the first 10 members, but if there are not 10 members it will return the whole set. A range of {5 6} would return the last two members. The default end option is the end of the set, so this could be written as {5}. If the start value is negative, the range is interpreted as being relative to the end of the set, so a range of { -2 -1 } would also specify the last two elements (as would just {-2}).

EXAMPLES

In the following example, the **get_clock_tree_pins** finds sink pins clock tree called clk1.

```
prompt> set sss [get_clock_tree_pins -filter is_sink -clock clk1]
{DFF4/CP DFF3/CP DFF2/CP DFF1/CP DFF0/CP DFF7/CP DFF6/CP DFF5/CP clk1iv1/Z
CTS_BUF_R__14911490/Z CTS_BUF_R__14871486/Z CTS_BUF_R__14861485/Z
clk1iv4/Z CTS_BUF_14841483/Z AFF1/CP AFF3/CP AFF6/CP AFF7/CP CP FF36/CP
FF35/CP FF34/CP FF33/CP FF32/CP FF31/CP FF46/CP FF45/CP FF44/CP
FF43/CP FF42/CP FF41/CP FF40/CP FF39/CP FF54/CP FF53/CP FF52/CP
FF51/CP FF50/CP FF49/CP FF48/CP FF47/CP FF62/CP FF61/CP FF60/CP FF59/CP FF58/CP ...}
```

In the following example, the **get_clock_tree_pins** finds ICG pins on the clock tree named clk1.

```
prompt> set sss [get_clock_tree_pins -filter is_on_icg -clock clk1]
{icg1/CK  icg1/ENCK }
```

This command can be used with the **change_selection** command to highlight parts of the clock tree. Here the command highlights any sinks 8 or more down.

```
prompt> change_selection \
  [get_clock_tree_pins -filter is_sink&&depth_max>=8]
1
```

The -expect option can be used as a simple validation check. This command checks for synthesized delay buffers with 200ps or more of cumulative delay. If such a structure is found, it raises a Tcl exception; otherwise, it is silent. For example, the following command raises an exception if any pin in the tree has a latency of 0.200 time units or greater.

```
prompt> get_clock_tree_pins -quiet -expect 0 -filter latency_max>0.200
1
```

In some cases, it might be useful to identify pins where an attribute is undefined, such as the skew at a pin with no balanced sinks below it. For example, in a simple, well-defined clock tree, all drive pins should have a defined, non-negative skew. This test finds any that did not.

```
prompt> get_clock_tree_pins -value_for_undefined -1 \
  -filter is_net_driver&&skew_max==undefined
```

If you need to access the skew, depth, height, or any other ephemeral attribute value after the command has executed, you can do so by sorting on that attribute and using the **-assign_to_variable** option. Each of the pins in the final sort list will correspond to one entry in this Tcl variable, and you can access it with the standard Tcl array syntax.

```
prompt> set slowest_pin [get_clock_tree_pins -filter is_sink \
  -sort_by latency_max -index_range -3 -assign_to_variable lat_values]
{block1/reg17/clk block1/reg52/clk block2/reg203/clk}
prompt> echo $lat_values(block1/reg52/clk)
0.0142
```

To get the pins of skew groups based on their ICG output pins, use the **-groups_from** option. In this example, the list returned as been indented to highlight the fact that it is a list of two collections.

```
prompt> set skew_groups_list [get_clock_tree_pins -filter is_sink \
  -groups_from { icg1.out  icg2.out} ]
{SUB2/FF0/CP SUB2/FF1/CP SUB3/FF0/CP SUB3/FF1/CP SUB4/FF0/CP SUB4/FF1/CP
SUB4/SUB5/FF0/CP SUB4/SUB5/FF1/CP SUB4/SUB6/FF0/CP SUB4/SUB6/FF1/CP
SUB7/FF0/CP SUB7/FF1/CP}
{SUB1/FF0/CP SUB1/FF1/CP SUB1/FF2/CP SUB1/FF3/CP SUB1/FF4/CP SUB1/FF5/CP
SUB1/FF6/CP SUB1/FF7/CP SUB1/FF8/CP SUB1/FF9/CP}

prompt> query [lindex $skew_groups 1]
{SUB1/FF0/CP SUB1/FF1/CP SUB1/FF2/CP SUB1/FF3/CP SUB1/FF4/CP SUB1/FF5/CP
SUB1/FF6/CP SUB1/FF7/CP SUB1/FF8/CP SUB1/FF9/CP}
```

The **-metrics** option can be used to gather many metrics at the same time. You can use Tcl to extract the values for further processing.

```
## gather some power related metrics and put them in a variable called "subs"
prompt> set met_names "height_max, subtree_delay_longest, \
  subtree_num_repeaters_max, subtree_total_area_max, \
  subtree_total_switching_power_max, toggle_rate_max"
prompt> get_clock_tree_pins -metrics $met_names \
  -filter is_net_driver -assign subs -clock CORE_CLK
...

## pick up the first element of the array
prompt> echo lindex [array names subs] 0] \
  $subs([lindex [array names subs] 0])
SUB1/FF5/latch/Q 2 3.2 3.0 3 21.23 35442112.22 0.4444
```

Note that with some Tcl code you can collate the elements in the lists returned with the "met_names" option string. You can also convert the output to CSV format and write it into a file, as shown in the following example:

```
echo  pin_name, $met_names > subs.csv
foreach indx [array names subs] {
 echo  -n $indx >> subs.csv
 foreach e $subs($indx) {
    echo -n  ", $e" >> subs.csv
 }
 echo  "" >> subs.csv
}
```

To get the hierarchical pins in a design, you must use the **-scan_all_hierarchical_pins** option.

```
prompt> get_clock_tree_pins -scan_all_hierarchical_pins \
  -filter is_on_physical_boundary

prompt> get_clock_tree_pins -statistics stats \
  -metrics latency_max,latency_min -sort_by depth_max
SFF6/CP SFF7/CP U02/B U27/D0 U28/A U28/B U28/C U29/A U90/B U91/A
clk1 clk2 clk3 ...

prompt> lindex $stats 0
{expression   {depth_max}}   {count 174} {total 679} {mean 3.90229893}
{std_dev 1.74259639} {max 7} {min 0}

prompt> lindex $stats 1
{ expression {latency_max}}   {count 174} {total 133.08136945265142}
{mean 0.764835477} {std_dev 0.247443825} {max 0.99067997932434082} {min 0}

prompt> lindex $stats 2
{ expression {latency_min}}   {count 174} {total 129.66261979239789}
{mean 0.745187461} {std_dev 0.244880974} {max 0.9508100152015686} {min 0}
```

SEE ALSO

get_pins(2)
report_clock_timing(2)

report_clock_qor(2)

get_clocks

Creates a collection of clocks from the current design. You can assign these clocks to a variable or pass them into another command.

SYNTAX

collection **get_clocks**
[-design *design*]
[-mode *mode*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
patterns

Data Types

design	collection
mode	collection
expression	string
exact_count	int
minimum_count	int
patterns	list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-mode *mode*

Specifies the mode for finding objects. If this is not specified, objects will be found in the current mode.

-filter *expression*

Filters the collection with *expression*. For any clocks that match *patterns*, the expression is evaluated based on the clock's attributes. If the expression evaluates to true, the clock is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches clock names against patterns. Patterns can include the wildcard characters "*" and "?".

DESCRIPTION

The **get_clocks** command creates a collection of clocks in the current design that match certain criteria. The command returns a collection if any clocks match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

You can use the **get_clocks** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_clocks** result to a variable.

When issued from the command prompt, **get_clocks** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_clocks** provides a fast, simple way to display clocks in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_clocks** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page. In addition, see the man page for **all_clocks**, which also creates a collection of clocks.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-mode** option.

EXAMPLES

The following example applies the **set_max_time_borrow** command to all clocks in the design matching "PHI*".

prompt> **set_max_time_borrow 0 [get_clocks "PHI*"]**

The following example sets the variable clock_list to a collection of clocks that have period of 15.

prompt> **set clock_list [get_clocks * -filter "period==15"]**

SEE ALSO

all_clocks(2)
collections(2)
create_clock(2)
query_objects(2)
report_clocks(2)
shell.common.collection_result_display_limit(3)

get_command_hooks

Get registered hooks for this command

SYNTAX

string **get_command_hooks** *commandName*

string *commandName*

ARGUMENTS

commandName

Command to query

DESCRIPTION

The **get_command_hooks** command returns information about any hooks registered on the specified command. The data returned is in Tcl dict format and may be queried using that command.

EXAMPLES

When a command has both before and after hooks registered the data will look like this

```
prompt> get_command_hooks place_opt
before {before0 report_timing}
after {after0 {echo "Done with placement"}}
```

SEE ALSO

add_command_hook(2)
remove_command_hook(2)
get_current_hook_command(2)
dict(2)

get_command_option_values

Queries current or default option values.

SYNTAX

get_command_option_values
[-default | -current]

-command *command_name*

Data Types

command_name string

ARGUMENTS

-default

Gets the default option values, if available.

-current

Gets the current option values, if available.

-command *command_name*

Gets the option values for this command.

DESCRIPTION

This command attempts to query a default or current value for each option (of the command) that has default and/or current-value-tracking enabled. Details of how the option value is queried depend on whether one of the **-current** or **-default** options is specified (see below).

A "Tcl array set compatible" (possibly empty) list of option names and values is returned as the Tcl result. The even-numbered entries in the list are the names of options that were enabled for default-value-tracking or current-value-tracking and had at least one of these values set to a not-undefined value). Each odd-numbered entry in the list is the default or current value of the option name preceding it in the list.

Any options that were not enabled for either default-value-tracking nor current-value-tracking are omitted from the output list.

Similarly, options that were enabled for default-value-tracking or current-value-tracking, but for which no (not-undefined) default or current value is set, are omitted from the result list.

If neither **-current** nor **-default** is specified, then for each command option that has either default-value-tracking or current-value-tracking (or both) enabled, the value returned is as follows:

- The current value is returned if current-value-tracking is enabled and a (not-undefined) current value has been set;
- Otherwise the default value is returned if default-value-tracking is enabled and a (not-undefined) default value has been set;
- Otherwise the name and value pair for the option is not included in the result list.

If **-current** is specified, the value returned for an option is the current value if current-value-tracking is enabled, and a (not-undefined) current value has been set; otherwise the name and value pair for the option is omitted from the result list.

If **-default** is specified, the value returned for an option is the default value if default-value-tracking is enabled, and a (not-undefined) default value has been set; otherwise the name and value pair for the option is omitted from the result list.

The result list from **get_command_option_values** includes option values of both dash options and positional options (assuming that both kinds of options of a command have been enabled for value-tracking).

The command issues a Tcl error in a variety of situations, such as if an invalid command name was passed in with **-command**.

EXAMPLES

The following example shows the use of **get_command_option_values**:

```
prompt> test -opt1 10 -opt2 20
1

prompt> get_command_option_values -command test
-bar1 10 -bar2 20
```

SEE ALSO

preview(2)
set_command_option_value(2)

get_connected_routing

Creates a collection of net shapes and vias that are physically connected to the specified collection of objects on the same nets. The specified objects can be of type net, shape, via, pin, terminal, and port.

SYNTAX

collection **get_connected_routing**
[-hierarchical]
[-cross_net]
[*objects*]

Data Types

objects collection

ARGUMENTS

-hierarchical

Traces connections across hierarchical module boundaries. For example, all net shapes of a connected feedthrough net are included in result collection when this option is specified.

-cross_net

Traces physically connected shapes across different nets, except PG nets. For example, short circuit net shapes will be included in the results.

DESCRIPTION

This command creates a collection of net shapes and vias by tracing the physical connectivity of the specified objects. If no such objects are specified as command arguments, the selected objects in layout view will be used. Routing corridor shapes are also supported, but are treated specifically. For them, this command creates a collection of routing corridor shapes that are physically connected and belong to the same routing corridor. The **-hierarchical** and **-cross_net** command options have no effect in this case.

EXAMPLES

The following example gets the connected routing objects of the selected net shapes.

```
prompt> change_selection [get_shapes PATH_31_0]
prompt> get_connected_routing
```

The following example creates a collection of shapes and vias that connect to the specified hierarchical net of the specified shape.

```
prompt> get_connected_routing -hierarchical [get_shapes PATH_31_0]
```

The following example creates a collection of the short circuit shapes and vias that connect to the specified shape.

```
prompt> get_connected_routing -cross_net [get_shapes PATH_31_0]
```

SEE ALSO

change_selection(2)
get_shapes(2)
gui_select_connected_net_shapes(2)

get_constraint_groups

Returns a collection of constraint groups from the current design.

SYNTAX

collection **get_constraint_groups**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[-type string]
[*patterns* | -of_objects *of_objects*]

Data Types

design collection
expression string
exact_count int
minimum_count int
of_objects list
patterns string

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with *expression*. For any constraint group that match the patterns option, the expression is evaluated based on the constraint group's attributes. If the expression evaluates to *true*, the constraint group is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for constraint groups level-by-level relative to the current instance. This option is useful only with *patterns* and not with **-of_objects**.

-type *string*

Specifies the type of constraint groups to be searched. By default all types of constraint groups are searched.

-of_objects *of_objects*

Creates a collection containing the constraint groups of the specified nets and bundles. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches constraint group names against the *patterns* argument. The *patterns* argument can include the wildcard character "*".

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

DESCRIPTION

The **get_constraint_groups** command creates a collection of constraint groups from the current design that match certain criteria. The command returns a collection of constraint groups if any constraint group matches the criteria. If no objects match the criteria, the empty string is returned.

You can use the **get_constraint_groups** command at the command prompt, or you can nest it as an argument to another command,

such as **report_attributes**. In addition, you can assign the **get_constraint_groups** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a collection that contains all the constraint groups in the current design.

```
prompt> get_constraint_groups
{differential_pair_1 shielding_2 net_priority_3}
```

The following example returns a collection of all the constraint groups tied to a net.

```
prompt> get_constraint_groups -of_objects net12
{shielding_2 net_priority_3}
```

SEE ALSO

create_wire_matching(2)
create_length_limit(2)
create_differential_group(2)
create_net_shielding(2)
create_net_priority(2)
create_bus_routing_style(2)
collections(2)
query_objects(2)
shell.common.collection_result_display_limit(3)

get_core_area

Creates a collection containing the core area of the current design.

SYNTAX

collection **get_core_area**
[-design design]

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

DESCRIPTION

This command returns a collection containing the core area of the current design. If the core area is not defined, the command returns an empty string.

The core area is a box that contains all rows. It represents the placeable area for standard cells. The core area is typically smaller than the cell boundary. Pads, I/O pins, and top-level power and ground rings are typically found outside the core area. Standard cells, macros, and wire tracks are typically found inside the core area. There should be only one core area in a design.

The core_area object is a first class object,and it has multiple attributes, such as bbox, cell_id, name and so on.You can use **list_attributes** and **report_attributes** to browse attributes.You can use **get_attribute** and **lindex** to get coordinates of the current design die area.

EXAMPLES

The following example gets the core area of the current design.

```
prompt> get_core_area
{unit_core}
```

The following example shows how to list the attributes of the core_area class by using **list_attributes** command.

```
prompt> list_attributes -application -class core_area
```

The following example shows how to get the attribute values of the core_area object by using **report_attributes** command.

```
prompt> report_attributes -application -class core_area [get_core_area]
```

The following example shows how to get coordinates of the current design die area,by using **get_attribute** command.

```
prompt> get_attribute [get_core_area] bbox
{0.000 0.000} {300.480 396.620}
```

SEE ALSO

get_attribute(2)
list_attributes(2)

get_corners

Creates a collection of corners in the current design. You can assign these corners to a variable or pass them into another command.

SYNTAX

collection **get_corners**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *of_objects*]

Data Types

design collection
expression string
exact_count int
minimum_count int
of_objects list
patterns list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with *expression*. For any objects that match, the expression is evaluated based on the object's attributes. If the expression evaluates to true, the object is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *of_objects*

Creates a collection containing the corners of the specified modes. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches corner names against patterns. Patterns can include the wildcard characters "*" and "?".

DESCRIPTION

The **get_corners** command creates a collection of corners matching *patterns* in the current design and which pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Multicorner-Multimode Support

This command works on all corners.

SEE ALSO

all_corners(2)
collections(2)
create_corner(2)
current_corner(2)

get_cputime

Retrieves the overall user time associated with the current process.

SYNTAX

float **get_cputime**
  [-all]
  [-format]

string *format*

ARGUMENTS

all

If specified then it gives cpu time for this process and it's children.

format

This argument takes a *printf* style formatting string for a floating point number.

DESCRIPTION

The *get_cputime* command returns the accumulated user time, in seconds, for the current process. If the *format* string is omitted, an integer number of seconds is returned. If the format string is specified, a floating point number is returned. The number includes fractions of a second elapsed and is formatted in accordance with given specifier. Refer to the Unix man page for *printf* for a detailed description of how to format a floating point argument.

```
prompt> get_cputime "%g"
3.74
```

SEE ALSO

get_mem(2)

get_cross_probing_info

Gets cross-probing information about cells, ports, modules and blocks and returns the results as a string.

SYNTAX

string **get_cross_probing_info**
[-unique_source]
objects

Data Types

objects collection

ARGUMENTS

-unique_source

Prevents duplicate source file and line number combinations from being returned in the result.

If you specify this option, the string that is returned follows a different format; it does not use a pair of braces between the different source positions of an object or the results between different objects. Instead, the tool returns the results in the following format:
obj1_file1:obj1_line1 obj2_file1:obj2_line1 obj2_file2:obj2_line2
If an object has no known source position or valid file and line number, nothing is returned for the object.

objects

Specifies the cell, port, module and block objects to query.

DESCRIPTION

This command provides cross-probing information about specified cells, ports, modules and blocks and returns the results as a string.

You must have a design that was analyzed and elaborated or synthesized using this tool. The tool provides information about where the cell or port or module or block was created, specifying which file and line number the object came from. Some objects in the design might have been merged, which can result in an object with multiple source locations. In this case, multiple results are returned for the object.

The results of each object are separated by a pair of braces between each source position and between each object. For example, if you specify two objects with the **get_cross_probing_info** command, obj1 and obj2, where obj1 has one source position and obj2 has two source positions, the tool returns the results in the following format:

{ { obj1_file1:obj1_line1 } } { { obj2_file1:obj2_line1 } {obj2_file2:obj2_line2 } }

You can prevent duplicate file and line number combinations in the results by using the **-unique_source** option. In this case, the tool returns the results in the following format:

obj1_file1:obj1_line1 obj2_file1:obj2_line1 obj2_file2:obj2_line2

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows results from the **get_cross_probing_info** command:

```
prompt> get_cross_probing_info [get_cells -hierarchical]
{ {/usr/joe/design/m.v:21} } { {/usr/joe/design/m.v:22} } { {/usr/joe/design/m.v:21} }
{ {/usr/joe/design/m.v:22} } { {/usr/joe/design/m.v:28} } { {/usr/joe/design/m.v:12} }
{ {/usr/joe/design/m.v:13} } { {/usr/joe/design/m.v:14} } { {/usr/joe/design/m.v:4} }
```

The following example shows results from the **get_cross_probing_info** command with the **-unique_source** option:

```
prompt> get_cross_probing_info [get_cells -hierarchical] -unique_source
/usr/joe/design/m.v:21 /usr/joe/design/m.v:22 /usr/joe/design/m.v:28 /usr/joe/design/m.v:12
/usr/joe/design/m.v:13 /usr/joe/design/m.v:14 /usr/joe/design/m.v:4
```

The following example shows results from the **get_cross_probing_info** command for modules:

```
prompt> get_cross_probing_info [get_modules]
{ {/usr/joe/design/top.v:1} } { {/usr/joe/design/top.v:14} } { {/usr/joe/design/top.v:32} }
```

The following example shows results from the **get_cross_probing_info** command for modules with the **-unique_source** option:

```
prompt> get_cross_probing_info [get_modules] -unique_source
/usr/joe/design/top.v:1 /usr/joe/design/top.v:14 /usr/joe/design/top.v:32
```

The following example shows results from the **get_cross_probing_info** command for blocks:

```
prompt> get_cross_probing_info [get_blocks]
{ {/usr/joe/design/top.v:1} }
```

The following example shows results from the **get_cross_probing_info** command for blocks with the **-unique_source** option:

```
prompt> get_cross_probing_info [get_blocks] -unique_source
/usr/joe/design/top.v:1
```

SEE ALSO

report_cross_probing(2)

cross_probing_filter(2)

get_current_checkpoint

Get information about the current active checkpoint.

SYNTAX

string **get_current_checkpoint**
[-name]
[-position]
[-script]

Data Types

ARGUMENTS

-name

Returns the name of the current checkpoint name

-position

Returns the name of the stage depending on if the execution is currently at before/after/replace.

-script

Returns the name of the checkpointed TCL script

DESCRIPTION

Provides introspection into the current checkpoint. This command must be called with one and only one of the -name, -position, and -script options. This command should be executed within a checkpoint, either directly in the block of Tcl enwrapped by the eval_checkpoint command, or indirectly within a checkpoint action or report.

The -name option returns the name of the current checkpoint. A typical application is to use the checkpoint name as part of the filename when writing out reports to disk from a checkpoint report. This ensures unique names for the reports and identifies at which checkpoint that report was generated.

The -position option returns the position within the current checkpoint. While executing a before action or report, it returns 'before'. While executing the checkpointing block of Tcl commands, or substituting it with a replacement action, it returns 'self'. While executing an after action or report, it returns 'after'. A typical application is to use the position in a filename written to disk by a checkpoint report. This ensures that if a report is associated to run before and after the same checkpoint, it writes to two unique filenames.

The -script option returns the full block of Tcl command/s enwrapped by the eval_checkpoint statement. A typical application is to use this command information in a replacement action, for example to modify the arguments passed to a Tcl command within the checkpoint.

EXAMPLES

This example defines a checkpoint report to write report_qor to a file on disk. It then associates this report to run before and after the place_opt checkpoint. When the place_opt checkpoint runs, two report_qor files are written with unique names by using the get_current_checkpoint -name and -position options.

```
## In ./checkpoint.config.tcl
create_checkpoint_report qor {
 set name [get_current_checkpoint -name]
 set pos [get_current_checkpoint -position]
 redirect -file ./checkpoint/$name.$pos.qor.rpt { report_qor }
}

associate_checkpoint_report -enable qor -before place_opt
associate_checkpoint_report -enable qor -after place_opt

## In the flow script place_opt.tcl
eval_checkpoint place_opt { place_opt }

## Writes two files:
## ./checkpoint/place_opt.before.qor.rpt
## ./checkpoint/place_opt.after.qor.rpt
```

This example defines a checkpoint action run high-effort congestion-driven placement on all create_placement* and refine_placement* checkpoints. get_current_checkpoint -command is used to access and modify the original command options to force high congestion effort. associate_checkpoint_action uses the -replace option to replace the original checkpoint command contents with the modified command contents.

```
## In ./checkpoint.config.tcl
 create_checkpoint_action high_eff_cong {
 eval [get_current_checkpoint -command] -congestion -congestion_effort high
 }
associate_checkpoint_action -enable high_eff_cong -replace "create_placement* refine_placement*"

## In the flow script place_opt.tcl
eval_checkpoint create_placement { create_placement -timing_driven }

## Upon execution, that create_placement command will instead run:
create_placement -timing_driven -congestion -congestion_effort high
```

SEE ALSO

create_checkpoint_action(2)
remove_checkpoint_actions(2)
create_checkpoint_report(2)
remove_checkpoint_reports(2)
associate_checkpoint_action(2)
associate_checkpoint_report(2)
set_checkpoint_options(2)
eval_checkpoint(2)
reset_checkpoints(2)
get_checkpoint_data(2)

get_current_ems_database

Returns a collection containing the current EMS database.

SYNTAX

collection **get_current_ems_database**

ARGUMENTS

The **get_current_ems_database** command takes no arguments.

DESCRIPTION

The **get_current_ems_database** command returns the current EMS database as part of a collection. An EMS database is called "current", if it is opened in memory and is the currently active database.

EXAMPLES

The following example returns a collection containing the current EMS database.

```
prompt> create_ems_database test.ems
prompt> get_current_ems_database
{test.ems}
```

SEE ALSO

close_ems_databases(2)
create_ems_database(2)
get_ems_databases(2)
open_ems_database(2)
save_ems_database(2)
set_current_ems_database(2)

get_current_hook_command

Get the currently running command string

SYNTAX

string **get_current_hook_command**

ARGUMENTS

DESCRIPTION

The **get_current_hook_command** command may only be run from within a command hook. The command returns the command string used to invoke the command.

Note that option values may have been abbreviated when the hooked command was executed. In that case the information returned will also include abbreviated option values.

SEE ALSO

add_command_hook(2)
remove_command_hook(2)
get_command_hooks(2)

get_current_mismatch_config

Obtains the current mismatch config.

SYNTAX

string **get_current_mismatch_config**

ARGUMENTS

The **get_current_mismatch_config** command has no arguments.

DESCRIPTION

The **get_current_mismatch_config** command returns the name of the current config in the session. By default, a session has **default** config as current config. Use command **set_current_mismatch_config** to change current config.

EXAMPLES

The following example gets the name of the current config:

```
prompt> get_current_mismatch_config
default


prompt> set_current_mismatch_config auto_fix
1


prompt> get_current_mismatch_config
auto_fix
```

SEE ALSO

report_design_mismatch(2)
get_mismatch_types(2)
get_mismatch_objects(2)
set_current_mismatch_config(2)
create_mismatch_config(2)
report_mismatch_configs(2)

get_defined_attributes

Returns attribute names currently defined for the specified class.

SYNTAX

string **get_defined_attributes**
-class *class_name*
-return_classes
[-details]
[-application]
[-user]
[*attr_pattern*]

Data Types

class_name string
attribute_names list
data_type string

ARGUMENTS

-class *class_name*

Specifies the class for which to get the attributes. Valid classes are application defined. Note legal with the *-return_classes* option.

-return_classes

Return the set of application defined classes. Not legal with the *-class* option.

-details

Gets additional information on a single attribute.

-application

Returns application-defined attributes only. This option is mutually exclusive with the *-user* option.

-user

Returns user-defined attributes only. This option is mutually exclusive with the *-application* option.

attr_patterns

Specifies a list of attribute names or glob style patterns to check on the existences of.

DESCRIPTION

This command returns a list of attribute names that are defined for the specified class.

There are three legals forms for this command:

- get_defined_attributess -return_classes

Returns the application defined object classes

- get_defined_attributes -class cell [<pattern>] Returns all the attributes matching the optional pattern
- get_defined_attributes -class cell -details <attrName> Returns more detailed information about the attribute. See below for details

  When **-details** is specified, the return value is a Tcl list that is suitable as input to the Tcl **array set** command. The returned value is a list with an even number of arguments. Each odd-numbered element in the list is a key, and each even-numbered element in the list is the value of the previous key.

  The supported keys are as follows:

name

This key contains the name of the attribute

infor

The short help defined for the attribbute

type

The type of value the attribute storesn

settable

The value is settable with the set_attribute command.

application

The value is 1 if the attribute is application defined, else 0.

subscripted

The value is 1 if the attribute is subscripted, else 0.

min_value

Not always present. If present value is the minimum allowed value.

max_value

Not always present. If present value is the maximum allowed value.

allowd_values

Not always present. If present value is the set of allowed values

allowed_subscripts

Present if the attribute is subscripted. Contains list of legal subcripts for the attribute if the attribute uses global subscripts. If empty, then the subscripts vary per object.

smart_subscripts

Present if the attribute is subscripted. Contains the list of supported smart subscripts if the attribute supports smart subscripts.

EXAMPLES

The following are examples of the **get_defined_attributes** command:

```
prompt> get_defined_attributes -class cell
name full_name object_class ....

prompt> get_defined_attributes -class cell *name*
name full_name reference_name ...

prompt> get_defined_attributes -class cell name
name name info {The simple name of the cell} type string application 1
settable 0 subscripted 0
```

SEE ALSO

get_attribute(2)
help_attributes(2)

get_defined_commands

Get information on defined commands and groups.

SYNTAX

string **get_defined_commands** [-details]
  [-groups]
  [*pattern*]

string *pattern*

ARGUMENTS

-details

Get detailed information on specific command or group.

-groups

Search groups rather than commands

pattern

Return commands or groups matching pattern. The default value of this argument is "*".

DESCRIPTION

The **get_defined_commands** gets information about defined commands and command groups. By default the command returns a list of commands that match the specified pattern.

When **-details** is specified, the return value is a list that is suitable as input to the **array set** command. The returned value is a list with an even number of arguments. Each odd-numbered element in the list is a key name, and each even-numbered element in the list os the value of the previous key. The **-details** option is only legal if the pattern matches exactly one command or group.

When **-group** is specified with **-details**, the supported keys are as follows:

name

This key contains the name of the group.

info

This key contains the short help for the group.

commands

This key contains the commands in the group

When **-details** is used with a command, the supported keys are as follows:

name

This key contains the name of the command.

info

This key contains the short help for the command.

groups

This key contains the group names that this command belongs to.

options

This key contains the options defined for the command. The value is a list.

return

This key contains the return type for the command.

Each element in the *options* list also follows the key value pattern. The set of available keys for options are as follows:

name

This key contains the name of the option.

info

This key contains the short help for the option.

value_info

This key contains the short help string for the value

type

The type of the option.

required

The value will be 0 or 1 depending if the option is optional or required.

is_list

Will be 1 if the option requires a list.

list_length

If a list contains the list length constraint. One of any, even, odd, non_empty or a number of elements.

allowed_values

The allowed values if the option has specified allowed values.

min_value

The minimum allowed value if the option has specified one.

max_value

The maximum allowed value if the option has specified one.

EXAMPLES

prompt> **get_defined_commands *collection**
add_to_collection append_to_collection copy_collection filter_collection
foreach_in_collection index_collection sort_collection
prompt> **get_defined_commands -details sort_collection**
name sort_collection info {Create a sorted copy of the collection}
groups {} options {{name -descending info {Sort in descending order}
value_info {} type boolean required 1 is_list 0} {name -dictionary info
{Sort strings dictionary order.} value_info {} type boolean required 1
is_list 0} {name collection info {Collection to sort} value_info
collection type string required 0 is_list 0} {name criteria info {Sort
criteria - list of attributes} value_info criteria type list required 0
is_list 1 list_length non_empty}}

SEE ALSO

help(2)
man(2)

get_density_rules

Creates a collection by selecting density rules from a lib or tech.

SYNTAX

collection **get_density_rules**
[-design *design*]
[-library *library*]
[-tech *tech*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
patterns | -of_objects *of_objects*

Data Types

design collection
library collection
expression string
exact_count int
minimum_count int
of_objects list
patterns string

ARGUMENTS

-design *design*

Specifies the block in which to find density rules.

If you do not specify a block or technology data, the command searches for density rules in the current block.

-library *library*

Specifies the library in which to find density rules. The command searches the technology data of the specified library to find density rules.

If you do not specify a block or technology data, the command searches for density rules in the current block.

-tech *tech*

Specifies the technology data in which to find density rules.

If you do not specify a block or technology data, the command searches for density rules in the current block.

-filter *expression*

Filters the collection with *expression*. For any density rule that matches the patterns option, the expression is evaluated based on the density rule's attributes. If the expression evaluates to *true*, the density rule is included in the result. By default, this option is off.

Use the **list_attributes** command to determine the density rule attributes.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive. **-nocase** and **-exact** are mutually exclusive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *of_objects*

Creates a collection containing the density rules associated with the specified objects. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches the density rule names against the patterns in the specified block or technology data.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

DESCRIPTION

This command creates a collection of density rules by selecting density rules from the specified block or technology data that meet the selection criteria. It returns a collection if one or more density rules meet the selection criteria. If no density rules match the selection criteria, it returns an empty string.

Use the **get_density_rules** command as an argument to another command or assign its result to a variable.

See the **collections** man page for information about working with collections.

EXAMPLES

The following example creates a collection of all density rules defined in the current block that are associated with objects whose name starts with VIA_C:

```
prompt> get_density_rules -of_objects VIA_C*
{FATVIA12_230_530 FATVIA12_530_230}
```

The following example creates a collection of all density rules from the technology data named test with a maximum density limit of 100:

```
prompt> get_density_rules -tech test -filter "max_density == 100"
{DENSITY_RULE_0 DENSITY_RULE_2 DENSITY_RULE_4 DENSITY_RULE_6 DENSITY_RULE_8
DENSITY_RULE_10 DENSITY_RULE_12 DENSITY_RULE_14 DENSITY_RULE_16}
```

SEE ALSO

report_density_rules(2)
get_attribute(2)
query_objects(2)

get_design_checks

Creates a collection of both user-defined and pre-defined checks.

SYNTAX

collection **get_design_checks**
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *of_objects*]

Data Types

expression string
exact_count int
minimum_count int
of_objects list
patterns string

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*, which must be a boolean expression based on check attributes. You can determine check attributes by using the **list_attributes** command and option **-class ems_check** to it. If the expression evaluates to **true**, the check is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this option is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *of_objects*

Takes a collection or list of objects which are checks.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches the check names in the current design against the specified patterns. You can specify the patterns by using the following formats:

(1) A collection of checks

(2) An asterisk (*), which indicates all checks

(3) Specific check names

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

DESCRIPTION

This command returns a collection of both pre-defined and user-defined checks that helps the user in knowing which checks are currently available.

EXAMPLES

(1) The following example shows how get_design_checks is used to create a list of available checks.

```
prompt> get_design_checks

   { routes mib_alignment pg_drc pg_missing_vias pin_placement
    analyze_design_violations rp_constraints finfet_grid cts_qor
```

feedthroughs pre_route_stage legality scan_chain design_mismatch
design_extraction clock_trees unbound pre_clock_tree_stage
pre_placement_stage timing mv_design routability }

(2) The following example shows how to get a collection of available checks ending with "_stage".

prompt> **get_design_checks *_stage**

{ pre_placement_stage pre_clock_tree_stage pre_route_stage }

SEE ALSO

list_attributes(2)
report_attributes(2)
create_check_design_strategy(2)
report_check_design_strategy(2)
check_design(2)

get_design_rules

Returns a collection of design rules from the technology data.

SYNTAX

collection **get_design_rules**
[-tech *tech_object*]
[-library *library*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *of_objects*]

Data Types

tech_object collection
library string or collection
expression string
exact_count int
minimum_count int
of_objects list
patterns string

ARGUMENTS

-tech *tech_object*

Specifies the technology data in which to search for design rules.

By default, the command searches for design rules in the technology data of the current library.

The **-tech** and **-library** options are mutually exclusive; you can specify only one.

-library *library*

Specifies the library in which to search for design rules. The command searches the technology data of the specified library to find design rules.

By default, the command searches for design rules in the technology data of the current library.

The **-tech** and **-library** options are mutually exclusive; you can specify only one.

-filter *expression*

Filters the collection with *expression*. For any design rule that matches the patterns option, the expression is evaluated based on the design rule's attributes. If the expression evaluates to *true*, the design rule is included in the result.

-quiet

Suppresses warning and error messages if no object matches. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *of_objects*

Creates a collection containing the design rules of the specified techs. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches design rule names against the *patterns* argument. The *patterns* argument can include the wildcard character "*".

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

DESCRIPTION

The **get_design_rules** command creates a collection of design rules from the tech that match certain criteria. The command returns a collection of design rules if any design rule matches the criteria. If no objects match the criteria, the empty string is returned.

You can use the **get_design_rules** command at the command prompt, or you can nest it as an argument to another command, such as **report_attributes**. In addition, you can assign the **get_design_rules** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a collection that contains all the design rules in the tech of current library.

```
prompt> get_design_rules
{DESIGN_RULE_0 DESIGN_RULE_1 DESIGN_RULE_2 DESIGN_RULE_3 DESIGN_RULE_4 DESIGN_RULE_5}
```

SEE ALSO

create_pr_rule(2)
remove_pr_rules(2)
report_design_rules(2)

get_designs

Creates a collection of one or more designs in memory. You can assign these designs to a variable or pass them into another command.

SYNTAX

collection **get_designs** [-hierarchical]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
patterns

Data Types

expression string
exact_count int
minimum_count int
patterns list

ARGUMENTS

-hierarchical

Searches for designs inferred by the design hierarchy relative to the current instance. The full name of the object at a particular level must match the patterns. The use of this option does not force an auto link.

-filter *expression*

Filters the collection with *expression*. For any designs that match *patterns*, the expression is evaluated based on the design's attributes. If the expression evaluates to true, the design is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are

mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches design names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type design.

DESCRIPTION

The **get_designs** command creates a collection of designs from those currently loaded into memory that match certain criteria. The command returns a collection if any designs match the *patterns* and pass the filter (if specified). If no objects matched your criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_designs** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_designs** result to a variable.

When issued from the command prompt, **get_designs** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_designs** provides a fast, simple way to display designs in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_designs** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the designs that begin with 'mpu.' Although the output looks like a list, it is just a display. A complete listing of designs is available using the **list_blocks** command.

```
prompt> get_designs mpu*
{"mpu_0_0", "mpu_0_1", "mpu_1_0", "mpu_1_1"}
```

The following example shows that, given a collection of designs, you can remove those designs.

```
prompt> remove_block [get_designs mpu*]
Removing design mpu_0_0...
Removing design mpu_0_1...
Removing design mpu_1_0...
Removing design mpu_1_1...
```

SEE ALSO

collections(2)
filter_collection(2)
list_blocks(2)
query_objects(2)
regexp(2)
remove_blocks(2)
shell.common.collection_result_display_limit(3)

get_dff_connections

Get dff_connection objects representing the results of **compute_dff_connections**.

SYNTAX

status **get_dff_connections**
[-filter *expression*]
[-max_reg *register_count*]
[-max_gate *gate_count*]

[-from *list_of_connection_points*]
[-to *list_of_connection_points*]
[-through *list_of_connection_points*] [-through *list_of_connection_points*] [...]

[-no_from_to_through]

[-quiet]
[*path pattern list*]

Data Types

expression string
register_count int
gate_count int
list_of_connection_points collection or list of names of ports, pins, blocks, module boundaries or virtual groups. Connection points can be ports, pins (hierarchical or not), cells (blocks), module boundaries or virtual module boundary groups. These can be specified as a collection or a name, such as *-from "BlOCK_1"* or *-from "BLOCK_1/bus1*"*. Any cells/modulle boundaries specified (without a pattern specifying specific pins) will be turned into a list of ALL pins belonging to that object.

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any dff_connection objects that match *patterns* or *objects*, the expression is evaluated based on the attributes of the dff_connection. If the expression evaluates to true, the dff_connection is included in the result.

-max_reg *register_count*

Specifies the maximum number of registers a path can contain and still be returned. It is an error to specify a larger value than

used during **compute_dff_connections**.

-max_gate *gate_count*

Specifies the maximum number of gates a path can contain and still be returned. It is an error to specify a larger value than used during **compute_dff_connections**.

-from *list_of_connection_points*

Return only paths starting at one of the specified connection points. See Data Types.

-to *list_of_connection_points*

Return only paths ending at one of the specified connection points. See Data Types.

-through *list_of_connection_points*

Return only paths going through one of the specified connection points. See Data Types. *-through* can be specified more than once. If *-through* is specified more than once, then a path must match one connection point from the each of the *-through* lists in order.

By default, start and end points of a path are also searched when *-through* is specified (see next option).

See **NOTES** and **KNOWN LIMITATIONS** for more information about *-through*.

-no_from_to_through

Do not check start or end points of the dff_connection against *-through* patterns. The default is to check start and end points against *-through* patterns.

-quiet

Do not report errors/warnings when the patterns specified for *-from*, *-to* or *-through* do not match any netlist objects.

path pattern list

A list of patterns used to match the name of paths. Only paths matching one or more of the patterns are returned. If not specified, this filter is inactive.

DESCRIPTION

This command returns dff_connection objects representing paths from the Data Flow Flylines (DFF) tracer (**compute_dff_connections**). For the attributes of dff_connection objects, see dff_connection_object_attributes(3).

By default, all connections are returned. The returned paths can be filtered using **-from**, **-to**, **-through**, **-no_from_to_through** and **[path pattern list]** as well as the standard collection filtering option **-filter**. .

Calling *compute_dff_connections* will invalidate existing dff_connection objects/collections.

The results of *get_dff_connections* with various block/macro/bound/pin options should closely match those of *report_dff_connections*. However, be aware that for blocks and bounds, *report_dff_connections* does the equivalent of a *-through* search, not *-from* or *-to*. Since the tracer stops at macros and ports, paths generally stop there, so *-through* is not needed if trying to do a *-from* or *-to* equivalent search.

NOTES

The *-through* option can be very useful. However, since the entire contents of the **detail** attribute of each path must be searched

(and potentially **start** and **end** fields), it is definitely slower than *-from* and *-to*. Exact performance of *-through* depends directly on the number of objects in the **detail** attribute. Multiple *-through* options will not increase this penalty significantly, since the detail field of each dff_connection is searched only once in order to match all *-through* arguments.

KNOWN LIMITATIONS

There is one significant limitation to *-through*. Due to the tracer technology, the **detail** field contains actual pin instances that correspond to the entries of the **hierarchy_pins** field. All other entries in the detail field are **cells**, NOT **pins**. Searching for a path that goes through a particular pin of a specific register will fail, EVEN if the **detail** field contains that register, since some objects will have multiple pins. Therefore, *-through* can be used to search for specific hierarchy pins (or their parent cells) and non-hierarchical cells only. Specifying non-hierarchical pins to *-through* will fail to match anything.

EXAMPLES

The following example traces the current design using default constraints, and then the first call to *get_dff_connections* gets paths starting from a block BLOCK_A. The second example assumes *compute_dff_connections* was already called, and gets paths ending at any port.

```
prompt> compute_dff_connections
prompt> get_dff_connections -from "BLOCK_A"

prompt> set port_list [get_ports]
prompt> get_dff_connections -to $port_list
```

SEE ALSO

collections(2)
compute_dff_connections(2)
dff_connection_object_attributes(3)
filter_collection(2)
report_dff_connections(2)

get_dft_hierarchical_pins

Returns a collection of ports and hierarchical pins created during DFT insertion.

SYNTAX

collection **get_dft_hierarchical_pins**

ARGUMENTS

The **get_dft_hierarchical_pins** command has no arguments.

DESCRIPTION

This command returns a collection of ports and hierarchical pins created during DFT insertion command in the current design.

EXAMPLES

The following example shows how to use the **get_dft_hierarchical_pins** command to show that the queried objects also have the attribute "created_during_dft":

```
prompt> set ports [ get_dft_hierarchical_pins ]
{mid1/test_se mid1/bot1/test_se mid1/bot1/c1/test_se mid1/test_si mid1/bot1/test_si mid1/bot1/c1/test_si}

prompt> foreach_in_collection p $ports {puts [get_attribute $p created_during_dft]}
true
true
true
true
true
true
```

SEE ALSO

current_design(2)
insert_dft(2)

get_domain_elements

Creates a collection of elements of power domains.

SYNTAX

collection **get_domain_elements**
[*power_domains*]

Data Types

power_domains list

ARGUMENTS

power_domains

Returns a collection of domain elements associated with the specified power domains. If no power domains are specified, the tool returns the domain elements of all power domains.

DESCRIPTION

The **get_domain_elements** command creates a collection of elements of power domains in the current design.

Until commit_upf command is executed and power intent is finalized, power domains of subblocks are not propagated and will not be included in the result.

EXAMPLES

The following example creates a power domain, then queries the elements of the power domain.

```
prompt> create_power_domain SUB_DOMAIN -elements [get_cells mid1]
1
prompt> get_domain_elements SUB_DOMAIN
{mid1}}
```

SEE ALSO

create_power_domain(2)
report_power_domains(2)

get_drc_error_data

Creates a collection of DRC error data objects.

SYNTAX

collection **get_drc_error_data**
[-all]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-of_objects *objects*]
[*patterns*]

Data Types

expression string
exact_count int
minimum_count int
objects list
patterns list

ARGUMENTS

-all

Includes unopened DRC error data objects, if matched. The default is to return only DRC error data objects that are currently open.

-filter *expression*

Filters the collection with *expression*.

For any DRC error data objects that match the *patterns* or *objects* argument, the expression is evaluated based on the DRC error data object's attributes. If the expression evaluates to true, the DRC error data object is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* argument and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters.

The **-regexp** and **-exact** options are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find.

If the command finds a different number of objects, it raises a Tcl error and stops processing.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find.

If the command finds fewer objects, it raises a Tcl error and stops processing.

-expect_each_pattern_matches

Specifies that each pattern in *patterns* must match at least one object.

If one or more patterns does not match, the command raises a Tcl error and stops processing.

-of_objects *objects*

Creates a collection of DRC error data objects connected to the specified objects. You can specify the following types of objects: DRC error types (drc_error_type), DRC errors (drc_error), and blocks.

The **-of_objects** option and *patterns* argument are mutually exclusive; you can specify only one. If you do not specify either, the wildcard "*" is used as the *pattern*.

patterns

Matches DRC error data object names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of DRC error data objects.

The **-of_objects** option and *patterns* argument are mutually exclusive; you can specify only one. If you do not specify either, the wildcard "*" is used as the *pattern*.

DESCRIPTION

The **get_drc_error_data** command creates a collection of DRC error data objects that match certain criteria. The command returns a collection if any DRC error data objects match the *patterns* or *objects* and pass the filter (if specified). If no objects match the

criteria, the command returns the empty string.

To include DRC error data objects that are not open, use the **-all** option. The **get_drc_error_data** command creates a collection that contains the DRC error data object, but does not open the DRC error data object. To query or modify the error type objects and error objects in the DRC error data object, you must open the DRC error data object with the **open_drc_error_data** command.

You can use the **get_drc_error_data** command at the command prompt or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_drc_error_data** result to a variable.

When issued from the command prompt, **get_drc_error_data** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by using the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of the **get_drc_error_data** command provides a fast, simple way to display the DRC error data objects in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use the **get_drc_error_data** command as an argument to the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a collection that contains the DRC error data object named "dppinassgn.err", and then opens it.

```
prompt> set data [get_drc_error_data -all dppinassgn.err]
{"dppinassgn.err"}
prompt> open_drc_error_data $data
{"dppinassgn.err"}
```

The following example creates a collection that contains the DRC error data object associated with the block named "my_design".

```
prompt> get_drc_error_data -of_objects my_design
{"dppinassgn.err"}
```

SEE ALSO

close_drc_error_data(2)
create_drc_error_data(2)
filter_collection(2)
open_drc_error_data(2)
query_objects(2)
remove_drc_error_data(2)
save_drc_error_data(2)
collections(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_drc_error_types

Creates a collection of physical DRC error types from the given error data. You can assign these physical DRC error types to a variable or pass them into another command.

SYNTAX

collection **get_drc_error_types**
-error_data *drc_error_data*
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns*]
[-of_objects *objects*]

Data Types

drc_error_data collection
expression string
exact_count int
minimum_count int
patterns list
objects list

ARGUMENTS

-error_data *drc_error_data*

Specifies the error data in which to find objects.

-filter *expression*

Filters the collection with *expression*. For any physical DRC error types that match *patterns* (or *objects*) the expression is evaluated based on the physical DRC error type's attributes. If the expression evaluates to true, the physical DRC error type is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a true regular expression rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with true regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. Use this option when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *objects*

Creates a collection of physical DRC error types connected to the specified objects. In this case, each object is either a named drc_error or a drc_error collection. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

patterns

Matches physical DRC error type names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of drc_error_type. *patterns* and **-of_objects** are mutually exclusive; you must specify one, but not both. If both the *patterns* and **-of_objects** are omitted, then the wildcard "*" is used as the *pattern*

DESCRIPTION

The **get_drc_error_types** command creates a collection of physical DRC error types in the given error data that match certain criteria. The command returns a collection if any physical DRC errors match the *patterns* or *objects* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_drc_error_types** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_drc_error_types** result to a variable.

When issued from the command prompt, **get_drc_error_types** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.commmon.collection_result_display_limit** application option.

The "implicit query" property of **get_drc_error_types** provides a fast, simple way to display physical DRC error types in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_drc_error_types** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the physical DRC error type that is named "route short". Although the output looks like a list, it is not. The output is just a display. The quotation marks around (route short) are required because the type name contains a space.

```
prompt> set data [open_drc_error_data -file_name my_design.err]
prompt> get_drc_error_types -error_data $data {"route short"}
{"route short"}
```

The following example shows that, given a collection of physical DRC errors, you can query the physical DRC error types connected to those errors.

```
prompt> set error [get_drc_errors -error_data $data 9]
{"9"}
prompt> query_objects [get_drc_error_types -error_data $data -of_objects $error]
{"route short"}
```

The following example gets the physical DRC errors associated with an error type.

```
prompt> get_drc_errors -error_data $data -of_objects \
      [get_drc_error_types -error_data $data {"route short"}]
{"9", "10", "11"}
1
```

SEE ALSO

collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_drc_errors

Creates a collection of physical DRC errors from the given error data. You can assign these physical DRC errors to a variable or pass them into another command.

SYNTAX

collection **get_drc_errors**
-error_data *drc_error_data*
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns*]
[-of_objects *objects*]
[-boundary *region*]

Data Types

drc_error_data collection
expression string
exact_count int
minimum_count int
objects list
region region
patterns list

ARGUMENTS

-error_data *drc_error_data*

Specifies the error data for finding objects.

-filter *expression*

Filters the collection with *expression*. For any physical DRC errors that match *patterns* (or *objects*) the expression is evaluated based on the physical DRC error's attributes. If the expression evaluates to true, the physical DRC error is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *objects*

Creates a collection of physical DRC errors connected to the specified objects. In this case, each object is either a named drc_error_type, net, pin, port, cell, pin guide, voltage area, or a collection of these objects. **-of_objects** and *patterns* are mutually exclusive; you may specify one, but not both.

-boundary *region*

Creates a collection of physical DRC errors with shapes within or touching the specified boundary shape. The region may be entered as a bounding box or as a list of coordinates. **-boundary** and **-filter can both be specified together.**

patterns

Matches physical DRC error names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type drc_error. *patterns* and **-of_objects** are mutually exclusive; you may specify one, but not both. *patterns* is also mutually exclusive with **-of_objects**; you may specify one, but not both. If both the *patterns* and **-of_objects** are ommitted, then the wildcard "*" is used as the *pattern*

DESCRIPTION

The **get_drc_errors** command creates a collection of physical DRC errors in the given error data that match certain criteria. The command returns a collection if any physical DRC errors match the *patterns* or *objects* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of

an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_drc_errors** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_drc_errors** result to a variable.

When issued from the command prompt, **get_drc_errors** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_drc_errors** provides a fast, simple way to display physical DRC errors in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_drc_errors** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the physical DRC errors that is associated with an error type named "route short". Although the output looks like a list, it is not. The output is just a display. Note the required quotes because the type name contains a space.

```
prompt> set data [open_drc_error_data -file_name my_design.err]
prompt> get_drc_errors -error_data $data -filter {type_name == "route short"}
{"9", "10", "11"}
```

The following example shows that, given a collection of error types, you can query the physical DRC errors connected to those error types.

```
prompt> set type [get_drc_error_types -error_data $data {{route short}}]
{"route short"}
prompt> query_objects [get_drc_errors -error_data $data -of_objects $type]
{"9", "10", "11"}
```

The following example gets the **objects** attribute for physical DRC error 9, which in this case are two nets associated with the error.

```
prompt> get_attribute [get_drc_errors -error_data $data 9] objects
{"F_12_", "F_11_"}
1
```

SEE ALSO

collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_eco_bus_buffer_patterns

Creates a collection of ECO bus buffer patterns. You can assign these ECO bus buffer patterns to a variable or pass them into another command.

SYNTAX

collection **get_eco_bus_buffer_patterns**
patterns

Data Types

patterns list

ARGUMENTS

patterns

Matches ECO bus buffer pattern names against patterns. Patterns can include the wildcard characters "*" and "?". Patterns can also include collections of type ECO bus buffer pattern.

DESCRIPTION

The **get_eco_bus_buffer_patterns** command creates a collection of ECO bus buffer patterns.

EXAMPLES

The following example returns ECO bus buffer patterns that begin with "o".

prompt> **get_eco_bus_buffer_patterns "o*"**
{o_p1 o_p2}

SEE ALSO

add_buffer_on_route(2)
create_eco_bus_buffer_pattern(2)
remove_eco_bus_buffer_patterns(2)
report_eco_bus_buffer_patterns(2)

get_edit_groups

Creates a collection by selecting edit groups from the current design.

SYNTAX

collection **get_edit_groups**
[-filter *expression*]
[-quiet]
[-design *design_name*]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

expression string
design_name string
exact_count integer
minimum_count integer
patterns list
objects list
point list
region list

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the edit group attributes. You can determine the edit group attributes by using the **list_attributes** command. If the expression evaluates to **true**, the edit group is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-design *design_name*

Specifies the design module in which to search for edit groups.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for edit groups level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at** but not with **-of_objects**.

patterns

Matches the edit group names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-of_objects *of_objects*

Creates a collection containing the edit groups that contain the specified objects. If the objects are edit groups, the command returns the edit groups which the objects contain. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-at *point*

Creates a collection that contains all edit_groups at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all edit_groups that are completely inside the specified region and doesn't include edit_groups which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all edit_groups that are completely inside the specified region and the edit_groups which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all edit_groups which are abut/touching from inside or outside to the specified region and the edit_groups whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of edit groups that meet the selection criteria. It returns a collection handle if one or more edit groups meet the selection criteria. If no edit groups match the selection criteria, it returns an empty string.

You can use the **get_edit_groups** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of edit_groups by matching the edit group name against the pattern.

```
prompt> get_edit_groups EDIT_GROUP*
{"EDIT_GROUP_1", "EDIT_GROUP_2", "EDIT_GROUP_3"}
```

The following example creates a collection of edit groups with attribute filtering.

```
prompt> get_edit_groups -filter {ungroup_on_remove == last}
{"EDIT_GROUP_2"}
```

The following example creates a collection of all edit groups.

```
prompt> get_edit_groups *
{"EDIT_GROUP_1", "EDIT_GROUP_2", "EDIT_GROUP_3", "my_top_edit_group"}
```

The following example creates a collection of all edit groups containing the cells "U33" and "U60".

```
prompt> get_edit_groups -of_objects "U33 U60"
{"EDIT_GROUP_1", "EDIT_GROUP_2"}
```

The following example creates a collection of all edit groups with names that begin with "ED".

```
prompt> get_edit_groups -expect_each_pattern_matches ED*
{"EDIT_GROUP_1", "EDIT_GROUP_2", "EDIT_GROUP_3"}
```

SEE ALSO

add_to_edit_group(2)
shell.common.collection_result_display_limit(3)
collections(2)
create_edit_group(2)
filter_collection(2)
query_objects(2)

regexp(2)
remove_edit_groups(2)
remove_from_edit_group(2)
report_edit_groups(2)

get_edit_setting

Returns common edit settings.

SYNTAX

string **get_edit_setting**
[-ignore_fixed]
[-ignore_locked]
[-keep_pin_on_edge]
[-self_intersection]
[-update_floorplan]
[-update_color_mask]
[-auto_display_hidden]
[-select_partial_object]
[-select_edge]
[-select_vertex]
[-select_center_line]
[-select_center_vertex]
[-expand_hit_macro_cell]
[-expand_hit_blockage]
[-expand_hit_constraint]
[-honor_ndr]
[-hierarchical_routing]
[-pin_layer_policy]
[-specified_pin_layer]

ARGUMENTS

-ignore_fixed

Return the setting for the ignore locked ort fixed objects flag. The option is synonym of **-ignore_locked** option.

-ignore_locked

Return the setting for the ignore locked and modify locked objects flag. Use The option is synonym of **-ignore_fixed** option.

-keep_pin_on_edge

Return the setting for keeping pins on the parent edge.

-self_intersection

Return the setting for allowing shape self intersection.

-update_floorplan

Return the setting for updating the floorplan when cell boundary is changed.

-update_color_mask

Return the setting for updating the color mask for a net shapes to match the tracks they are snapped to (if the track has a specified mask color).

-auto_display_hidden

Return the setting for displaying hidden objects automatically.

-select_partial_object

Return the setting for allowing selection of partial object.

-select_edge

Return the setting for allowing object edge selection.

-select_vertex

Return the setting for allowing object vertex selection.

-select_center_line

Return the setting for allowing wire center line selection.

-select_center_vertex

Return the setting for allowing wire center point selection.

-expand_hit_macro_cell

Return the setting for expand hit type macro cell.

-expand_hit_blockage

Return the setting for expand hit type blockage.

-expand_hit_constraint

Return the setting for expand hit type placement or routing constraint.

-honor_ndr

Return the setting for honoring nondefault width and spacing during editing.

-hierarchical_routing

Return the setting for treating hierarchical nets as same net for routing and tracing.

-pin_layer_policy

Return the setting for layer policy while moving pins on the edge.

-specified_pin_layer

Return the setting for pin layer when layer policy is set to 'specified'.

DESCRIPTION

This command return the current edit setting for the specified option. Use this command to query one setting at a time.

EXAMPLES

The following example returns the current edit setting for self intersection.

```
prompt> get_edit_setting -self_intersection
1
```

SEE ALSO

set_edit_setting(2)

get_editability

Returns the hierarchical edit control of specified blocks.

SYNTAX

list **get_editability**
[-blocks *list_of_blocks*]

Data Types

list_of_blocks string or collection

ARGUMENTS

-blocks *list_of_blocks*

The hierarchical edit control of specified blocks would be reported with respect to the current top-level context. If this option is not specified, the default behavior shall report the hierarchical editability of the current_block.

DESCRIPTION

This command returns a list containing the true/false value of hierarchical edit controls corresponding to each of the specified blocks. A true value means the block is editable and false value means the block is uneditable.

EXAMPLES

The following example gets the hierarchical edit control of a single block:

```
prompt> get_editability -blocks blk1
false
```

The following example gets the hierarchical edit control of multiple blocks:

```
prompt> get_editability -blocks [get_blocks {blk1 blk2 blk3}]
{false true false}
```

SEE ALSO

set_editability(2)
report_editability(2)

get_edrc_setting

Returns common Editor DRC settings.

SYNTAX

string **get_edrc_setting**
[-name *rule_name*]
[-rules]
[-enclosed_via_spacing_rule]
[-general_via_spacing_rule]
[-minimum_edge_rule]
[-minimum_length_and_area_rule]
[-rdl_acute_angle_rule]
[-rdl_right_angle_rule]
[-via_density_rule]
[-honor_ndr]
[-dpt_odd_cycle]
[-dpt_precolor]
[-end_of_line_spacing_rule]
[-via_enclosure_rule]
[-metal_width_rule]
[-metal_span_spacing_rule]
[-end_of_line_keepout_rule]
[-filter_same_net_spacing]
[-check_fill]
[-max_error_limit]
[-max_shape_limit]
[-max_processing_time]
[-show_error_browser]
[-check_drc]

Data Types

rule_name string

ARGUMENTS

-name *string*

Returns the current setting for the specified rule name.

-rules

Returns a list of rule name and value pairs.

-enclosed_via_spacing_rule

Returns the current setting for the enclosed via spacing rule.

-general_via_spacing_rule

Returns the current setting for the general via spacing rule.

-minimum_edge_rule

Returns the current setting for checking the following minimum edge rules: maximum total number of short edge, maximum total short edge length, convex-concave minimum edge length, adjacent minimum edge length and special notch.

-minimum_length_and_area_rule

Returns the current setting for checking the minimum length and general minimum area rules.

-rdl_acute_angle_rule

Returns the current setting for checking acute angle violations for RDL routes.

-rdl_right_angle_rule

Returns the current setting for checking right angle violations for RDL routes.

-via_density_rule

Returns the current setting for checking the via density rule.

-honor_ndr

Returns the current setting for checking the following nondefault routing rules: metal spacing and via spacing.

-dpt_odd_cycle

Returns the current setting for checking the existence of DPT odd cycles.

-dpt_precolor

Returns the current setting for checking DPT spacing violation for colored shapes.

-end_of_line_spacing_rule

Returns the current setting for checking the following end-of-line spacing rules: end-of-line to end-of-line, one-neighbor end-to-end table-based, one-neighbor end-of-line and two-neighbor end-of-line.

-via_enclosure_rule

Returns the current setting for checking the following via enclosure rules: minimum enclosure, zero-neighbor end-of-line enclosure, and T-shape metal extension end-of-line enclosure.

-metal_width_rule

Returns the current setting for checking the following metal width rules: discrete metal width rule and signal routing maximum metal width rule.

-metal_span_spacing_rule

Returns the current setting for checking the following metal span spacing rules: metal span spacing and metal span orthogonal spacing rule.

-end_of_line_keepout_rule

Returns the current setting for the end of line keepout rule.

-filter_same_net_spacing

Skips spacing violations between shapes of the same net.

-check_fill

Returns the current setting for checking shapes inside fill cells.

-max_error_limit

Returns the current setting for the maximum number of DRC errors to report. The default is 200.

-max_shape_limit

Returns the current setting for the maximum number of shapes for DRC checking. The default is 400.

-max_processing_time

Returns the maximum time (in seconds) to allow for DRC checking.

-show_error_browser

Returns the current setting for whether to show the error browser when errors are detected.

-check_drc

Returns whether DRC checking is enabled for interactive editing.

DESCRIPTION

This command queries additional rules and settings for Editor DRC. By default, Editor DRC checks shorts and the following spacing rules: minimum spacing, via corner spacing, u-shape spacing, fat metal spacing and minimum width, net-type based spacing and via farm spacing.

EXAMPLES

This command queries the current setting for DPT odd cycle.

```
prompt> get_edrc_setting -name {DPT Odd Cycle}
1
```

SEE ALSO

set_edrc_setting(2)

get_ems_databases

Returns a collection of all opened EMS databases in memory.

SYNTAX

collection **get_ems_databases**
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *objects*]

Data Types

expression string
exact_count int
minimum_count int
patterns list
objects list

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the EMS database attributes. You can determine the EMS database attributes by using the **list_attributes** command and **-application -class ems_database** options. If the expression evaluates to **true**, the EMS database is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Treats the *patterns* argument as real regular expressions rather than simple wildcard patterns. The **-regexp** option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Specifies that matching is case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches the EMS database names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

Matches EMS database names against patterns. Patterns can include the wildcard characters the * (asterisk) and ? (question mark) wildcard characters or regular expressions, depending on the use of the **-regexp** option. For more details about using and escaping wildcards, see the **wildcards** man page. The **-of_objects** and *patterns* options are mutually exclusive; you can use only one. By default, the command uses * (asterisk) as the *pattern*.

-of_objects *objects*

Creates a collection containing the EMS databases associated with the specified objects. -of_objects works with objects of type ems_message.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

DESCRIPTION

The **get_ems_databases** command creates a collection of EMS databases that match certain criteria. The command returns a collection handle (identifier) if any EMS databases match the value of the *patterns* option and pass the filter (if specified). If no objects match the criteria, the command returns false.

You can use the **get_ems_databases** command at the command prompt or nest it as an argument to another command. You can also assign the **get_ems_databases** result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example uses a regular expressions to get a collection of EMS databases that end with *_0.ems:

```
prompt> get_ems_databases -regexp *_0.ems
{EMS_DB_0.ems}
```

SEE ALSO

close_ems_databases(2)
create_ems_database(2)
get_current_ems_database(2)
open_ems_database(2)
save_ems_database(2)
set_current_ems_database(2)

get_ems_rules

Returns a collection of EMS rules.

SYNTAX

collection **get_ems_rules**
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *objects*]

Data Types

expression string
exact_count int
minimum_count int
patterns list
objects list

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the EMS rule attributes. You can determine the EMS rule attributes by using the **list_attributes** command and **-application -class ems_rule** options. If the expression evaluates to **true**, the EMS rule is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Treats the *patterns* argument as real regular expressions rather than simple wildcard patterns. The **-regexp** option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Specifies that matching is case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches the EMS rule names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

Matches EMS rule names against patterns. Patterns can include the wildcard characters the * (asterisk) and ? (question mark) wildcard characters or regular expressions, depending on the use of the **-regexp** option. For more details about using and escaping wildcards, see the **wildcards** man page. The **-of_objects** and *patterns* options are mutually exclusive; you can use only one. By default, the command uses * (asterisk) as the *pattern*.

-of_objects *objects*

Creates a collection containing the EMS rules associated with the specified objects.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

DESCRIPTION

The **get_ems_rules** command creates a collection of EMS rules that match certain criteria. The command returns a collection handle (identifier) if any EMS rules match the value of the *patterns* option and pass the filter (if specified). If no objects match the criteria, the command returns false.

You can use the **get_ems_rules** command at the command prompt or nest it as an argument to another command. You can also assign the **get_ems_rules** result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example uses a regular expressions to get a collection of EMS rules that begin with EMS:

```
prompt> get_ems_rules EMS*
{EMS-001 EMS-002 EMS-003 EMS-004 EMS-005}
```

SEE ALSO

create_ems_rule(2)
edit_ems_rule(2)
report_ems_rules(2)
write_ems_rules(2)
remove_ems_rules(2)

get_equivalent_power_domains

Create a collection of power domains that are equivalent and satisfy the criteria to share a voltage area.

SYNTAX

status **get_equivalent_power_domains**
power_domain
[-functional_equivalence]

Data Types

power_domains string

ARGUMENTS

power_domain

Returns a collection of power domains that are equivalent and satisfy the criteria to share a primary voltage area the with the specified power domain.

-functional_equivalence

Specifies that domain supplies are considered as equivalent based on functional equivalence. If this options is not specified, by default, domain supplies are considered as equivalent based on electrical equivalence.

DESCRIPTION

The **get_equivalent_power_domains** command create a collection of equivalent power domains in the current design that can share a voltage area with the specified power domain.

This command supports the shared voltage area flow only with mv.upf.multi_pd_shared_voltage_area_flow set to new.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example queries the equivalent power domains to share a primary voltage area with the specified power domain.

```
prompt> get_equivalent_power_domains pd1
 {pd2 pd3}
```

SEE ALSO

report_power_domains(2)
create_voltage_area(2)
set_voltage_area(2)
check_equivalent_power_domains(2)

get_essential_points

Lists the essential points that influence a specified set of objects.

SYNTAX

int **get_essential_points**
[*object_list*]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]

Data Types

object_list list
mode_list list
corner_list list
scenario_list list

ARGUMENTS

-objects *object_list*

List of objects - pins/ports/nets/cells - for which to list the activity influencing essential points. For a net, we consider any essential point in its fanin. For pins/ports, we consider any essential point in the fanin; if the pin/port happens to be an essential point, we list the pin/port itself. For cells, we consider any essential point that is in the fanin of its input pins.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering on modes occurs.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If this option is not specified, no filtering occurs on corners.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Output of the command is generated separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options is specified, the command lists for the current scenario in the design.

DESCRIPTION

This command takes a set of objects and for each specified scenario, generates a list of the essential points that influence these objects. To filter for scenarios, the *-modes*, *-corners* and *-scenarios* options can be used.

EXAMPLES

The following example shows the usage of the **get_essential_points** command to get the essential points of all the nets in the design .

```
prompt> get_essential_points [get_nets -hier]
Information: Activity for scenario scenario1 was cached, no propagation required. (POW-005)
****************************************
Report : get_essential_points
Design : ChipTop
Version: P-2019.03-SP4-BETA
Date   : Tue Jul  9 03:47:37 2019
****************************************
Inferring activity for the scenario: scenario1
Mode       : func
Corner     : max_corner
Time Unit  : 1ns
Fastest clock: clk2 with period 1.0
------------------------------------------------------------------------------------------------------------------------
{{net in1} {Essential Points {in1}}
{{net out1} {Essential Points {reg10/QN}}
{{net test_in1} {Essential Points {test_in1}}
{{net test_in2} {Essential Points {test_in2}}
{{net clk1} {Essential Points {clk1}}
{{net clk2} {Essential Points {clk2}}
{{net clksel} {Essential Points {clksel}}
{{net n1} {Essential Points {reg1/QN}}
{{net n2} {Essential Points {reg2/QN}}
{{net n3} {Essential Points {reg3/QN}}
{{net n4} {Essential Points {reg4/QN}}
{{net n5} {Essential Points {reg5/QN}}
{{net n6} {Essential Points {reg6/QN}}
{{net n7} {Essential Points {reg7/QN}}
{{net n8} {Essential Points {reg8/QN}}
{{net n9} {Essential Points {reg9/QN}}
```

In the following example, the **get_essential_points** command is used to query the essential point for a specified pin.

```
prompt> get_essential_points [get_pins reg5/D]
****************************************
Report : get_essential_points
Design : ChipTop
Version: P-2019.03-SP4-BETA
Date   : Tue Jul  9 03:51:11 2019
****************************************
Inferring activity for the scenario: scenario1
Mode       : func
```

```
Corner      : max_corner
Time Unit   : 1ns
Fastest clock: clk2 with period 1.0
-------------------------------------------------------------------------------------------------------------------
{{pin reg5/D} {Essential Points {reg4/QN}}
```

SEE ALSO

infer_switching_activity(2)
report_essential_points(2)
report_switching_activity(2)
report_activity(2)

get_estimated_wirelength

Estimates the net wire length by using a virtual router.

SYNTAX

float **get_estimated_wirelength**
[-nets *nets*]

Data Types

nets collection

ARGUMENTS

-nets *nets*

Specifies a list of nets for wire length estimation. By default, wire length estimation is performed for all nets in the current design and the command returns the total wire length.

DESCRIPTION

The **get_estimated_wirelength** command returns the estimated wire lengths for the nets. It uses a fast virtual router to build Steiner trees for the nets, taking consideration of the standard cell pins and block pins connected to them. The sum of the edge lengths of the Steiner trees are used as estimates for the wire lengths. The virtual router is multilevel physical hierarchy aware and blockages aware. However, the virtual router is not congestion aware and does not build an internal congestion map. Note that the **get_estimated_wirelength** command does not guarantee an accurate wire length calculation, rather it is just a quick estimation of the wire length.

For each given net, the wire length is calculated for the whole net as it traverses across all the hierarchies from top to bottom. Use the **set_editability** command to disable one or more blocks (may or may not include the top block itself) for planning. If a block is disabled, then the wire length estimation for the net segments inside this block is skipped. However, if a disabled block contains a sub-block that is still enabled for planning, then the wire length estimation for the net segments inside this planning enabled sub-block is still performed. With properly set **set_editability**, you can include or exclude specific net segments when creating the wire length estimate.

If a net connects to a block pin or a top port without a corresponding physical pin shape, the command ignores wire length estimates for the net segments connecting to this block pin or top port. The command issues a warning message to notify you of the problem.

If a net connects to a block pin or a top port that has multiple physical pin shapes, then the wire length estimation will randomly pick one. The command issues a warning message to notify you of the problem.

To get an accurate wire length estimation, the blocks in the top design should be a design view. If the blocks are with other views, such as an abstract view, the wire length estimation can be very different from the design view estimate, due to missing cells in the abstract view. The tool issues a warning message to notify you that net segments inside abstract views were encountered.

EXAMPLES

The following command performs wire length estimation for net **n1**.

```
prompt> get_estimated_wirelength -nets n1
```

SEE ALSO

place_pins(2)
set_editability(2)

get_exception_groups

Creates a collection of exception_groups from a collection of exceptions. You can assign the result collection to a variable or pass them into another command.

SYNTAX

collection **get_exception_groups**
[-filter *expression*]
collection1

Data Types

expression string
collection1 collection

ARGUMENTS

collection

Specifies the collection of exceptions to be filtered.

-filter *expression*

Filters the collection with the *expression* option. For any exception_groups that match the *patterns* option, the expression is evaluated based on the exception_group's attributes. If the expression evaluates to true, the exception_group is included in the result. If this option is not give, then all exception_groups in *collection* will be added to the result collection.

DESCRIPTION

The **get_exception_groups** command creates a collection of exception_groups from the *collection1* collection. The command returns a collection containing all exception_groups matching the *expression* argument of the -filter option.

You can use the **get_exception_groups** command at the command prompt, or you can nest it as an argument to another command. In addition, you can assign the **get_exception_groups** result to a variable.

For information about getting collection of exceptions, see the **get_exceptions** man page.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command works on the mode of the specified exceptions.

EXAMPLES

The following command sequence first get a collection of exceptions with from pins "**a***". The result collection is assigned to a variable named **exp_obj**.

The second command then, from **exp_obj**, gather a collection of exception_groups with **type** equal to "through". The result is assigned to a variable named **through_groups_clct**.

```
prompt> set exp_obj [get_exceptions -from a*]
prompt> set through_groups_clct \
  [get_exception_groups $exp_obj -filter "type==through"]
```

SEE ALSO

get_exceptions(2)
shell.common.collection_result_display_limit(3)
list_attributes(2)
report_attributes(2)
exception_group_attributes(3)

get_exceptions

Creates a collection of exceptions in the current scenario. You can assign these exceptions to a variable or pass them into another command.

SYNTAX

collection **get_exceptions**
[-design *design*]
[-from *from_list*
  | -rise_from *rise_from_list*
  | -fall_from *fall_from_list*]
[-through *through_list*]*
[-rise_through *rise_through_list*]*
[-fall_through *fall_through_list*]*
[-to *to_list*
  | -rise_to *rise_to_list*
  | -fall_to *fall_to_list*]
[-filter *expression*]
[-quiet]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]

Data Types

design collection
expression string
exact_count int
minimum_count int

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-from *from_list*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If you specify a cell, one path startpoint on that cell is affected. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from ***fall_from_list***

Same as the **-from** option, except that the path must fall from the objects specified.If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through ***through_list***

Specifies a list of pins, ports, and nets through which the paths must pass that are to be reset. Nets are interpreted to imply the leaf-level driver pins. If you omit **-through**, all timing paths specified using the **-from** and **-to** options are affected. You can specify **-through** more than once in one command invocation. For a discussion of the use of multiple **-through** options, see the DESCRIPTION section.

-rise_through ***rise_through_list***

This option is similar to the **-through** option, but applies only to paths with a rising transition at the through points. You can specify **-rise_through** more than once in one command invocation. For a discussion of the use of multiple **-through** options, see the DESCRIPTION section.

-fall_through ***fall_through_list***

This option is similar to the **-through** option, but applies only to paths with a rising transition at the through points. You can specify **-fall_through** more than once in one command invocation. For a discussion of the use of multiple **-through** options, see the DESCRIPTION section.

-to ***to_list***

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-rise_to ***rise_to_list***

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to ***fall_to_list***

Same as the **-to** option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-filter ***expression***

Filters the collection with *expression*. For any exceptions that match the from/through/to criteria, the expression is evaluated based on the exception's attributes. If the expression evaluates to true, the exception is included in the result.

-expect ***exact_count***

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least ***minimum_count***

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

DESCRIPTION

The **get_exceptions** command creates a collection of exceptions in the current scenario that match certain criteria. You can assign these exceptions to a variable or pass them into another command.

The command returns a collection if any exceptions match the from/through/to specifiers and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

You can use the **get_exceptions** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_exceptions** result to a variable.

When issued from the command prompt, **get_exceptions** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_exceptions** provides a fast, simple way to display exceptions in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_exceptions** as an argument to **query_objects**. Note that the name shown for an exception is a brief id for printing only. You cannot search for exceptions by this name.

For information about collections and the querying of objects, see the **collections** man page. In addition, see the man page for **all_exceptions**, which also creates a collection of exceptions.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example sets the variable e1 to a collection of false_path exceptions with OUT1 as a to_pin.

prompt> **set e1 [get_exceptions -to OUT1 -filter "type==false_path"]**

SEE ALSO

all_exceptions(2)
collections(2)
set_false_path(2)
set_multicycle_path(2)
set_max_delay(2)

set_min_delay(2)
shell.common.collection_result_display_limit(3)

get_fill_cells

Creates a collection of fill_cells. You can assign these fill_cells to a variable or pass them into another command.

SYNTAX

collection **get_fill_cells**
[-design *design*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[-filter *expression*]
patterns

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with *expression*. For any fill_cells that match *patterns* the expression is evaluated based on the fill_cell's attributes. If the expression evaluates to true, the fill_cell is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-hierarchical

Searches for fill_cells in the full **fill cell** hierarchy in the top-level design. By default, the search is limited to the top-level design.

Using this option allows the hierarchical fill data to be searched. It does not enable search across child physical blocks.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches fill_cell names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type fill_cell.

DESCRIPTION

The **get_fill_cells** command creates a collection of fill_cells in the current design, that match certain criteria. The command returns a collection if any fill_cells match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

If any *patterns* fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_fill_cells** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_fill_cells** result to a variable.

When issued from the command prompt, **get_fill_cells** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable **collection_result_display_limit**.

The "implicit query" property of **get_fill_cells** provides a fast, simple way to display fill_cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_fill_cells** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the fill cells in the current block.

```
prompt> get_fill_cells
{FILL_INST_0}
```

The following example queries the fill cells in the FILL_INST_0 instance whose names start with FILL_INST_2.

```
prompt> get_fill_cells -hierarchical FILL_INST_0/FILL_INST_2*
{FILL_INST_0/FILL_INST_2 FILL_INST_0/FILL_INST_20}
```

SEE ALSO

collections(2)
filter_collection(2)
remove_fill_cells(2)

get_flat_cells

Creates a collection of leaf cells that match certain criteria in the current design.

SYNTAX

collection **get_flat_cells**
[-design *design*]
[-all]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[-hsc *separator*]
patterns | -of_objects *objects*

Data Types

design collection
expression string
exact_count int
minimum_count int
separator string
patterns list
objects list

ARGUMENTS

-design *design*

Specifies the top-level block for finding objects.

The default is the current design.

-hierarchical

Expands the search expanded to the entire physical hierarchy.

By default, the command searches only in the current physical hierarchy.

-all

Includes physical-only cells in the collection. Physical cells are cells that either do not have any ports or have only power and ground ports.

By default, physical-only cells are not included in the collection.

-filter *expression*

Filters the collection with *expression*.

The command evaluates the cells that match *patterns* or *objects* based on the attributes of the cell. If the expression evaluates to true, the cell is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are never suppressed.

-regexp

Views the *patterns* argument as regular expressions rather than simple wildcard patterns.

This option also modifies the behavior of the =~ and !~ filter operators to compare with regular expressions rather than simple wildcard patterns.

The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Performs case-insensitive pattern matching.

-exact

Disables simple pattern matching.

Use this option to search for objects that contain the * and ? wildcard characters.

The **-exact** and **-regexp** options are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find.

If the command finds a different number of objects, it issues an error message and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find.

If the command finds fewer objects, it issues an error message and command processing stops.

-expect_each_pattern_matches

Ensures that each pattern in *patterns* matches at least one object.

If one or more patterns does not match, the command issues an error message and command processing stops.

-hsc *separator*

Specifies the hierarchical separator character.

The default is /. Valid values are /, @, ^, #, ., and |.

patterns

Matches cell names against patterns.

Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of cells.

-of_objects ***objects***

Creates a collection of leaf cells connected to or associated with the specified objects.

Each object in the list is a name, pattern, or collection. The objects can be a cell, pin, net, bound, io_guide, io_ring, voltage_area, voltage_area_shape, edit_group, rp_group, or power_strategy.

The **-of_objects** argument and *patterns* option are mutually exclusive; you must specify one, but not both.

In addition, you can use the **-hierarchical** option with the **-of_objects** option only if the objects are cells or rp_groups.

DESCRIPTION

The **get_flat_cells** command creates a collection of leaf cells from the current design that match the specified criteria. Leaf cells are those cells processed during place and route. They are shown and can be selected in the GUI layout window.

If the *patterns* or *objects* argument fails to match any objects and the design is not linked, the command automatically links the design.

The **get_flat_cells** and **get_cells** commands both return cell objects. The collections returned by both commands are interchangeable and can be passed along to any command that accepts cell objects. However, the **get_flat_cells** command differs from the **get_cells** command in several ways.

- The **get_cells** command performs the pattern search based on the logical hierarchy of the design and is consistent with the commands used for synthesis and timing analysis in the Design Compiler, IC Compiler, and PrimeTime tools.

  The **get_flat_cells** command searches all leaf cells to match the *patterns* argument on their full names relative to the current design; it does not depend on the current instance. It performs pattern search in a way that is consistent with the place-and-route view of the design, and thus is simpler for many tasks in the Fusion Compiler flow.

You can choose to use either command, depending on the type of search you are doing.

Regular expression matching is the same as in the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_flat_cells** command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the **get_flat_cells** result to a variable.

When entered at the command prompt, the **get_flat_cells** command behaves as though the **query_objects** command is called to display the objects in the collection. By default, a maximum of 100 objects are displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of the **get_flat_cells** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, to display the object class), use the **get_flat_cells** command as an argument to the **query_objects** command. For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following examples query all soft macro cells in the current design. Although the output looks like a list, it is only a text display.

```
prompt> get_flat_cells * -filter {is_soft_macro==true}
{I_ORCA_TOP/I_BLENDER_1 I_ORCA_TOP/I_BLENDER_2 I_ORCA_TOP/I_BLENDER_3
I_ORCA_TOP/I_BLENDER_4 I_ORCA_TOP/I_BLENDER_5}
```

The following examples query all black box cells in the current design. Although the output looks like a list, it is only a text display.

```
prompt> get_flat_cells * -filter {is_black_box == true }
{I_CLOCK_GEN/I_PLL_SD I_CLOCK_GEN/I_PLL_PCI
I_CLOCK_GEN/I_CLKMUL I_ORCA_TOP/I_BLENDER/latched_clk_en_reg I_ORCA_TOP/I_
CONTEXT_MEM/CONTEXT_RAM_3 I_ORCA_TOP/I_CONTEXT_MEM/CONTEXT_RAM_2 ...}
```

The following example queries the leaf cells whose full name begins with "BLOCK/". It includes all leaf cells any number of logical levels below the BLOCK hierarchical cell.

```
prompt> get_flat_cells BLOCK/*
{BLOCK/SB/U1 BLOCK/SB/U2 BLOCK/SB/U3 BLOCK/SB/U4 BLOCK/U5 BLOCK/U6
BLOCK/U7 BLOCK/U8 BLOCK/U9}
```

As a comparison to the previous example, the following example finds only cells that are direct children of the BLOCK hierarchical cell.

```
prompt> get_cells BLOCK/*
{BLOCK/SB BLOCK/U9 BLOCK/U5 BLOCK/U8 BLOCK/U7 BLOCK/U6}
```

The following example shows that, given a collection of nets, you can select the leaf cells connected to those nets. These cells are selected in a GUI layout view open for the current design.

```
prompt> set netsel [get_flat_nets r1_2_ ]
{r1_2_}
prompt> change_selection [get_flat_cells -of_objects $netsel ]
```

SEE ALSO

collections(2)
filter_collection(2)
get_cells(2)
get_flat_pins(2)
get_flat_nets(2)
list_attributes(2)
query_objects(2)
change_selection(2)
shell.common.collection_result_display_limit(3)
shell.dc_compatibility.black_box_is_leaf_cell(3)

get_flat_nets

Creates a collection of top-level nets of hierarchical net groups in the current design that match the specified criteria.

SYNTAX

collection **get_flat_nets**
[-design *design*]
[-all]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[-hsc *separator*]
patterns | -of_objects *objects*

Data Types

design string
expression string
exact_count integer
minimum_count integer
separator character
patterns list
objects list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. The default is the current design.

-all

Includes power and ground nets.

-filter *expression*

Filters the collection with *expression*. For any nets that match *patterns* (or *objects*), the expression is evaluated based on the attributes of the net. If the expression evaluates to true, the net is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are never suppressed.

-regexp

Considers the *patterns* argument as true regular expressions rather than simple wildcard patterns. This option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Performs case-insensitive pattern matching.

-exact

Disables simple pattern matching. Use this option to search for objects that contain the * and ? wildcard characters. The **-exact** and **-regexp** options are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, an error is reported and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, an error is reported and command processing stops.

-expect_each_pattern_matches

Ensures that each pattern in *patterns* matches at least one object. If one or more patterns does not match, an error is reported and command processing stops.

-hierarchical

Search will be expanded to multi-level physical hierarchy. By default, searches only in the current physical hierarchy.

-hsc *separator*

Specifies the hierarchical separator character. The default is /. Valid values are /, @, ^, #, ., and |.

patterns

Matches net names against patterns. Patterns can include the wildcard characters "*" and "?", or regular expressions, based on the **-regexp** option. Patterns can also include collections of type net. Patterns can also include subscript patterns, but not in conjunction with wildcard and regular expression pattern matching.

-of_objects *objects*

Creates a collection of nets connected to the specified objects. Each object is a named pin, a pin collection, a named cell, a cell collection, a named net or a net collection, or a routing corridor or a bundle.

The **-of_objects** and *patterns* arguments are mutually exclusive; you must specify one, but not both. In addition, **-hierarchical** and **-of_objects** can be used together only when **-physical_context** is also used and the *objects* argument to **-of_objects** are nets.

DESCRIPTION

The **get_flat_nets** command creates a collection of top-level nets of hierarchical net groups in the current design that match the specified criteria.

If any *patterns* (or *objects*) fail to match any objects and the current design is not linked, the design is automatically linked.

The **get_flat_nets** and **get_nets** commands both return net objects. The collections returned by both commands are interchangeable and can be passed along to any command that accept net objects. However, the **get_flat_nets** command differs from the **get_nets** command in several ways.

The **get_nets** command performs the pattern search based on the logical hierarchy of the design and is consistent with the commands used for synthesis and timing analysis in Design Compiler, IC Compiler, PrimeTime, and SDC. The **get_flat_nets** command searches the top-level nets of hierarchical net groups to match the *patterns* argument on their full names relative to the current design; it does not depend on the current instance. It performs pattern search in a way that is consistent with the place-and-route view of the design, and thus is simpler for many tasks in the Fusion Compiler flow. You can choose to use either command, depending on the type of search you are doing.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

Subscript patterns are also supported, but cannot be used in conjunction with wildcard or regular expression pattern matching.

Bus subscripting is specified with the format "*name*[*begin_index*:*end_index*:*step_n*]", and returns every nth bus net inclusively between the begin and end indexes. Note that the "[]" characters must be escaped because they are Tcl special characters (for example, A\[0:15:2\] or {A[0:15:2]}).

Name expansion subscripting is specified with the format "*name*(*begin_index*:*end_index*:*step_n*)", and returns every nth net inclusively between the begin and end indexes, which has a matching name without the delimiters characters (that is, net names do not actually contain the '(' and ')' characters). Note that ":*step_n*" is optional; the default step size is 1. Multiple "*begin_index*:*end_index*:*step_n*" patterns can be specified, separated with a ',' character.

You can use the **get_flat_nets** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_flat_nets** result to a variable.

When entered at the command prompt, **get_flat_nets** behaves as though **query_objects** is called to display the objects in the collection. By default, a maximum of 100 objects are displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_flat_nets** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, to display the object class), use **get_flat_nets** as an argument to **query_objects**. For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the top-level nets whose full name begins with "BLOCK/". Although the output looks like a list, it is just a text display.

```
prompt> get_flat_nets BLOCK/*
{BLOCK/SB/n76}
```

The following example shows that there is no top-level net whose full name begins with "BLOCK/" and ends with "n1".

```
prompt> get_flat_nets BLOCK/*n1
Warning: No nets matched 'BLOCK/*n1' (SEL-004)
```

As a comparison to the previous example, the following example shows that there is a net whose full name begins with "BLOCK/" and ends with "n1".

```
prompt> get_nets BLOCK/*n1
{BLOCK/n1}
```

The following selects the top-level nets of hierarchical net groups that connect to cells. You can see that these nets are selected in a GUI layout view open for the current design.

```
prompt> set cellsel [get_flat_cells I_ORCA_TOP/I_SDRAM_IF/U23276]
{I_ORCA_TOP/I_SDRAM_IF/U23276}

prompt> change_selection [get_flat_nets -of_objects $cellsel]
```

SEE ALSO

collections(2)
filter_collection(2)
get_flat_cells(2)
get_flat_pins(2)
get_nets(2)
list_attributes(2)
query_objects(2)
shell.common.collection_result_display_limit(3)

get_flat_pins

Creates a collection of leaf-cell pins that match certain criteria in the current design.

SYNTAX

collection **get_flat_pins**
[-design *design_name*]
[-all]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hsc *separator_character*]
[-hierarchical]
patterns | -of_objects *objects*

Data Types

design_name string
expression string
exact_count integer
minimum_count integer
separator_character string
patterns list
objects list

ARGUMENTS

-design *design_name*

Specifies the name of the design in which to find pins. The default is the current design.

-all

Includes power and ground pins.

-filter *expression*

Filters the collection with *expression*. For any pins that match *patterns* or *objects*, the expression is evaluated based on the attributes of the pin. If the expression evaluates to true, the pin is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are never suppressed.

-regexp

Views the *patterns* argument as true regular expressions rather than simple wildcard patterns. This option modifies the behavior of the =~ and !~ filter operators to compare with true regular expressions rather than simple wildcard patterns. The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Performs case-insensitive pattern matching.

-exact

Disables simple pattern matching. Use this option to search for objects that contain the * and ? wildcard characters. The **-exact** and **-regexp** options are mutually exclusive.

-expect *exact_count*

Specifies an exact number of objects to find. If the command finds a different number of objects, an error is reported and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, an error is reported and command processing stops.

-expect_each_pattern_matches

Ensures that each pattern in *patterns* matches at least one object. If one or more patterns does not match, an error is reported and command processing stops.

-hierarchical

Search will be expanded to multi-level physical hierarchy. By default, searches only in the current physical hierarchy.

-hsc separator_character

Specifies the hierarchical separator character. The default is /. Valid values are /, @, ^, #, ., and |.

patterns

Matches pin names against patterns. Patterns can include the wildcard characters "*" and "?", or regular expressions, based on the **-regexp** option. Patterns can also include collections of type pin. Patterns can also include subscript patterns, but not in conjunction with wildcard and regular expression pattern matching.

-of_objects *objects*

Creates a collection of pins connected to the specified objects. Each object is a named cell, net, cell collection, or pin collection. The **-of_objects** and *patterns* options are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_flat_pins** command creates a collection of pins on leaf cells from the current design that match the specified criteria. Leaf cells are those processed during place and route. They are shown and can be selected in the GUI layout window

If any *patterns* (or *objects*) fail to match any objects and the design is not linked, the design is automatically linked.

The **get_flat_pins** and **get_pins** commands both return pin objects. The collections returned by both commands are interchangeable and can be passed along to any command that accepts pin objects. However, the **get_flat_pins** command differs from the **get_pins** command in several ways.

The **get_pins** command performs the pattern search based on the logical hierarchy of the design and is consistent with the commands used for synthesis and timing analysis in Design Compiler, IC Compiler, PrimeTime, and SDC. The **get_flat_pins** command searches all pins of leaf cells to match the *patterns* argument on their full names relative to the current design; it does not depend on the current instance. It performs pattern search in a way that is consistent with the place-and-route view of the design, and thus is simpler for many tasks in the Fusion Compiler flow. You can choose to use either command, depending on the type of search you are doing.

When a cell has bus pins, **get_flat_pins** can find them in several ways. For example, if cell u1 has a bus "A" with indexes 2 to 0, and the design.bus_delimiters for your design is "%s[%d]", use "u1/A[*]" as the pattern. You can also find the same three pins with "u1/A" as the pattern.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

Subscript patterns are also supported, but cannot be used in conjunction with wildcard or regular expression pattern matching.

Bus subscripting is specified with the format "*name*[*begin_index*:*end_index*:*step_n*]", and returns every nth bus pin inclusively between the begin and end indexes. Note that the "[]" characters must be escaped because they are Tcl special characters (for example, A\[0:15:2\] or {A[0:15:2]}).

Name expansion subscripting is specified with the format "*name*(*begin_index*:*end_index*:*step_n*)", and returns every nth pin inclusively between the begin and end indexes, which has a matching name without the delimiters characters (that is, pin names do not actually contain the '(' and ')' characters). Note that ":*step_n*" is optional; the default step size is 1. Multiple "*begin_index*:*end_index*:*step_n*" patterns may be specified, separated with a ',' character.

You can use the **get_flat_pins** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_flat_pins** result to a variable.

When entered at the command prompt, **get_flat_pins** behaves as though **query_objects** in called to display the objects in the collection. By default, a maximum of 100 objects are displayed. You can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_flat_pins** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, to display the object class), use **get_flat_pins** as an argument to **query_objects**. For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the I pins of leaf cells that begin with "I_CLK". Although the output looks like a list, it is just a text display.

```
prompt> get_flat_pins I_CLK*/I
{I_CLK_SOURCE_SDRAM_CLK/I I_CLK_SOURCE_SYS_2x_CLK/I
I_CLK_SOURCE_SYS_CLK/I I_CLK_SOURCE_PCLK/I}
```

The following example selects pins of leaf cells whose name begins with "O_1". You can see that these pins are selected in a GUI layout view open for the current design.

```
prompt> set csel [get_flat_cells O_1*]
{O_10__reg O_11__reg O_12__reg O_13__reg O_1__reg}
prompt> change_selection [get_flat_pins -of_objects $csel]
```

SEE ALSO

collections(2)
filter_collection(2)
get_pins(2)
get_flat_cells(2)
get_flat_nets(2)
link_block(2)
list_attributes(2)
query_objects(2)
shell.common.collection_result_display_limit(3)
shell.dc_compatibility.black_box_is_leaf_cell(3)

get_generated_clock

Creates a collection of generated clocks.

SYNTAX

collection **get_generated_clocks**
[-design *design*]
[-mode *mode*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
patterns

Data Types

design	collection
mode	collection
expression	string
exact_count	int
minimum_count	int
patterns	list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-mode *mode*

Specifies the mode for finding objects. If this is not specified, objects will be found in the current mode.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

When combined with the *-regexp* option, makes matches case-insensitive.

-filter *expression*

Filters the collection with the *expression* option. For any generated clocks that match the *patterns* option, the expression is evaluated based on the generated clock's attributes. If the expression evaluates to true, the generated clock is included in the result.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches generated clock names against patterns. Patterns can include the wildcard characters "*" and "?".

DESCRIPTION

The **get_generated_clocks** command creates a collection of generated clocks from the current design that match certain criteria. The command returns a collection if any generated clocks match the *patterns* option and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

To create a generated clock in the design, use the **create_generated_clock** command. To remove a generated clock from the design, use the **remove_generated_clocks** command. To show information about clocks and generated clocks in the design, use the **report_clocks** command.

You can use the **get_generated_clocks** command at the command prompt, or you can nest it as an argument to another command (for example, the **query_objects** command). In addition, you can assign the **get_generated_clocks** command result to a variable.

When issued from the command prompt, the **get_generated_clocks** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of the **get_generated_clocks** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the the **query_objects** command, use the **get_generated_clocks** command as an argument to the **query_objects** command. For example, use this to display the object class.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-mode** option.

EXAMPLES

The following command applies the **set_clock_latency** command on all generated clocks in the design matching the "GEN*".

prompt> **set_clock_latency 2.0 [get_generated_clocks "GEN*"]**

The following command removes all the generated clocks in the design matching "GEN1*".

prompt> **remove_generated_clocks [get_generated_clocks "GEN1*"]**

SEE ALSO

collections(2)
create_generated_clock(2)
query_objects(2)
remove_generated_clocks(2)
report_clocks(2)
shell.common.collection_result_display_limit(3)

get_generated_clocks

Creates a collection of generated clocks.

SYNTAX

collection **get_generated_clocks**
[-design *design*]
[-mode *mode*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
patterns

Data Types

design collection
mode collection
expression string
exact_count int
minimum_count int
patterns list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-mode *mode*

Specifies the mode for finding objects. If this is not specified, objects will be found in the current mode.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

When combined with the *-regexp* option, makes matches case-insensitive.

-filter *expression*

Filters the collection with the *expression* option. For any generated clocks that match the *patterns* option, the expression is evaluated based on the generated clock's attributes. If the expression evaluates to true, the generated clock is included in the result.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches generated clock names against patterns. Patterns can include the wildcard characters "*" and "?".

DESCRIPTION

The **get_generated_clocks** command creates a collection of generated clocks from the current design that match certain criteria. The command returns a collection if any generated clocks match the *patterns* option and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

To create a generated clock in the design, use the **create_generated_clock** command. To remove a generated clock from the design, use the **remove_generated_clocks** command. To show information about clocks and generated clocks in the design, use the **report_clocks** command.

You can use the **get_generated_clocks** command at the command prompt, or you can nest it as an argument to another command (for example, the **query_objects** command). In addition, you can assign the **get_generated_clocks** command result to a variable.

When issued from the command prompt, the **get_generated_clocks** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of the **get_generated_clocks** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the the **query_objects** command, use the **get_generated_clocks** command as an argument to the **query_objects** command. For example, use this to display the object class.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-mode** option.

EXAMPLES

The following command applies the **set_clock_latency** command on all generated clocks in the design matching the "GEN*".

prompt> **set_clock_latency 2.0 [get_generated_clocks "GEN*"]**

The following command removes all the generated clocks in the design matching "GEN1*".

prompt> **remove_generated_clocks [get_generated_clocks "GEN1*"]**

SEE ALSO

collections(2)
create_generated_clock(2)
query_objects(2)
remove_generated_clocks(2)
report_clocks(2)
shell.common.collection_result_display_limit(3)

get_geometry_result

Returns a list of geometry-based rail results based on the type of result specified. The result is sorted by the voltage or resistance values in the order from high to low.

SYNTAX

list **get_geometry_result**
-type *result_type*
-net *net_name*
-touching *region*
[-layer *layer*]
[-top *top_num*]
[-threshold *threshold*]
[-percentage *percentage*]
[-histogram]

Data Types

result_type string
net_name string
layer string
region list
top_num integer
threshold float
percentage float

ARGUMENTS

-type *result_type*

Specifies the rail result type to report. The accepted types are voltage_drop and min_path_resistance. This option is a required option.

-net *net_name*

Specifies a power or ground net name for which rail results are to be returned. This option is a required option.

-layer *layer*

Specifies the layer name or layer number on which to return the rail result.

-touching *region*

Specifies a region such that geometries which are completely inside the specified region are returned with rail results. The region boundary can be a rectangle or a polygon. This is a required option to prevent returning all geometries in the whole design.

Users have to specify a region for the results to return.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

This option is mutually exclusive with the -top, -threshold, -percentage, or -histogram option.

-top ***top_num***

Returns the top number of geometries for the result type specified. This option is mutually exclusive with the -touching, -threshold, -percentage, or -histogram option.

-threshold ***threshold***

Returns geometries with the rail result value greater than the specified threshold. This option is mutually exclusive with the -touching, -top, -percentage, or -histogram option.

-percentage ***percentage***

Returns geometries with the rail result value such that its percentage ratio (calculated over the ideal supply voltage) is greater than the specified percentage. This option is mutually exclusive with the -touching, -top, -threshold, or -histogram option.

-histogram

Returns a histogram of rail results. The default histogram has 10 bins, which are expanded from the smaller value to the biggest value. This option is mutually exclusive with the -touching, -top, -threshold, or -percentage option.

The **get_geometry_result** command returns a list of geometry-based rail results. This command has to be run after the **analyze_rail** or **open_rail_result** command.

EXAMPLES

The following example returns a list of geometries within a rectangle area:

```
prompt> get_geometry_result -net VDD -type voltage_drop -touching {{1610 1968} {1620 1969}}
{ { 1616.475 1967.000 } { 1617.525 1969.800 } } metal5 0.00448
{ { 1608.810 1946.000 } { 1611.190 1974.000 } } metal7 0.00446
{ { 1608.000 1946.000 } { 1612.000 1974.000 } } metal9 0.00446
```

The following example returns the top 5 geometries on voltage drops within a rectangle region:

```
prompt> get_geometry_result -net VDD -type voltage_drop -top 5 -touching {0 0} {1000 1000}}
{ { 425.120 180.800 } { 432.720 181.860 } } M1 -0.0306
{ { 651.920 195.740 } { 659.120 196.800 } } M1 -0.0304
{ { 417.120 180.800 } { 425.120 181.860 } } M1 -0.03
{ { 432.720 180.800 } { 439.520 181.860 } } M1 -0.0297
{ { 409.120 180.800 } { 417.120 181.860 } } M1 -0.0287
```

SEE ALSO

analyze_rail(2)
open_rail_result(2)

get_grids

Creates a collection of grids from the current design.

SYNTAX

collection **get_grids**
[-filter *expression*]
[-quiet]
[-design *design_name*]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-type block | user]
patterns

Data Types

expression string
design_name string
exact_count integer
minimum_count integer
patterns list

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the grid attributes. You can determine the grid attributes by using the **list_attributes** command. If the expression evaluates to **true**, the specific grid is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-design *design_name*

Specifies the top design for finding objects. If this is not specified, objects are found in the current design.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~

and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-type block | **user**

Specifies the type of grid to return. This option can be used to get specific type of grids. It can take as value either *block*, for block grids or *user*, for user grids.

patterns

Matches the grid names against *patterns*. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

DESCRIPTION

This command creates a collection of grids from the current design that match certain criteria. The command returns a collection handle (identifier) if any grid matches the value of the *patterns* option and pass the filter (if specified). If no objects match the criteria, the command returns an empty string.

You can use the **get_grids** command at the command prompt or nest it as an argument to another command, such as **report_grids**. You can also assign the **get_grids** result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following returns a grid named gr1

prompt> **get_grids gr1**

{gr1}

The following returns all grids with names that begin with "gr"

prompt> **get_grids gr***
{gr1 gr2 gr3 gr4}

The following returns all block grids with name matching gr

prompt> **get_grids -type block gr***
{gr1 gr2}

The following returns all user grids with names that begin with "gr"

prompt> **get_grids -type user gr***
{gr3 gr4}

SEE ALSO

create_grid(2)
remove_grids(2)
report_grids(2)
set_block_grid_references(2)
set_grid(2)
snap_cells_to_block_grid(2)

get_groups

Creates a collection by selecting groups from the current design.

SYNTAX

collection **get_groups**
[-design *design_name*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hdl_of_module *module*]
[*patterns* | -of_objects *of_objects*]

Data Types

expression string
design_name string
exact_count integer
minimum_count integer
module string
patterns list
of_objects list

ARGUMENTS

-design *design_name*

Specifies the design in which to search for groups.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the group attributes. You can determine the group attributes by using the **list_attributes** command. If the expression evaluates to **true**, the group is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hdl_of_module

Matches the HDL groups contained in the specified module.

patterns

Matches the group names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Support hierarchical pattern when **-hdl_of_module** option is specified.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses asterisk (*) as the pattern.

-of_objects *of_objects*

Creates a collection containing the groups that contain the specified objects. If the objects are groups, the command also returns the groups which the objects contain. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

DESCRIPTION

This command creates a collection of groups that meet the selection criteria. It returns a collection handle if one or more groups meet the selection criteria. If no groups match the selection criteria, it returns an empty string.

You can use the **get_groups** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of groups by matching the group name against the pattern.

```
prompt> get_groups GROUP*
{"GROUP_1", "GROUP_2","GROUP_3" }
```

The following example creates a collection of groups with attribute filtering.

```
prompt> get_groups -filter {type == collection}
{"GROUP_2"}
```

The following example creates a collection of all groups.

```
prompt> get_groups
{"GROUP_1", "GROUP_2", "GROUP_3", "my_group"}
```

The following example creates a collection of all groups containing the shape "RECT_14_0".

```
prompt> get_groups -of_objects RECT_14_0
{"GROUP_1", "my_group"}
```

The following example creates a collection of all groups with names that begin with "GR".

```
prompt> get_groups -expect_each_pattern_matches GR*
{"GROUP_1", "GROUP_2", "GROUP_3"}
```

The following example creates a collection of HDL groups of module ORCA.

```
prompt> get_groups -hdl_of_module ORCA
{"hdl_blk_1", "hdl_blk_2","hdl_blk_3" }
```

The following example creates a collection of HDL group with the name HDL in the module ORCA.

```
prompt> get_groups -hdl_of_module ORCA HDL
{"HDL" }
```

SEE ALSO

add_to_group(2)
collections(2)
create_group(2)
filter_collection(2)
query_objects(2)
regexp(2)
remove_groups(2)
remove_from_group(2)
report_groups(2)
shell.common.collection_result_display_limit(3)

get_gui_stroke_bindings

Queries the data for stroke bindings.

SYNTAX

get_gui_stroke_bindings
[-dictionary *dict_name*]
[-builtin]

ARGUMENTS

-dictionary *dictionary_name*

Report only the bindings for the specified dictionary.

-builtin

Return information on the built-in commands that are available for use in bindings.

DESCRIPTION

This command queries the current state of stroke bindings for the application. If no options are specified, the application returns data for all dictionaries. The **-dictionary** option limits the data returned to only that for a given dictionary. The **-builtin** option queries for the built-in commands that are available for use with bindings.

A user does not typically use this command. The tool provides built-in reporting commands that produce human-readable reports from the information returned by this command. You can use the **report_gui_stroke_bindings** and **report_gui_stroke_builtins** commands for report generation.

This command returns its data as a Tcl list, suitable for further processing into reports or command streams by Tcl procedures.

DATA FORMAT

The command returns data differently depending on the options that are provided.

The data for a dictionary is a list of entries where each entry contains the stroke sequence and the data for the command invoked by that sequence. If the command queries more than one dictionary, another level of lists is added with dictionary name followed by dictionary data.

Built-in command data is formatted as a list of commands. Each command has the name of the built-in command followed by the data for the command.

EXAMPLES

The following example returns all binding information for all dictionaries.

```
psyn_gui-t> get_gui_stroke_bindings
```

The following example returns the bindings for the Graphics dictionary.

```
psyn_gui-t> get_gui_stroke_bindings -dictionary Graphics
```

The following example returns the built-in commands available for use in bindings.

```
psyn_gui-t> get_gui_stroke_bindings -builtin
```

SEE ALSO

report_gui_stroke_bindings(2)
report_gui_stroke_builtins(2)
set_gui_stroke_binding(2)
set_gui_stroke_preferences(2)

get_input_delays

Creates a collection of input delays for the current scenario. You can assign these input delays to a variable or pass them into another command.

SYNTAX

collection **get_input_delays**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
-of_objects *objects*
[-filter *expression*]
[-quiet]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]

Data Types

mode_list collection
corner_list collection
scenario_list collection
objects collection
expression string
exact_count int
minimum_count int

ARGUMENTS

-modes *mode_list*

Specifies the scenarios from which to retrieve the input delays. If this option is given, all scenarios of the specified modes will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners *corner_list*

Specifies the scenarios from which to retrieve input delays. If this option is given, all scenarios of the specified corners and the current mode will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios *scenario_list*

Specifies the scenarios from which to retrieve input delays. The **-modes** or **-corners** option must not be specified with this option.

-of_objects *objects*

Creates a collection of input delays defined on the specified objects. Each object is either a named pin, a pin collection, a named port or port collection.

-filter *expression*

Filters the collection with *expression*. For any objects that match, the expression is evaluated based on the object's attributes. If the expression evaluates to true, the object is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

DESCRIPTION

The **get_input_delays** command creates a collection of input delays matching *patterns* in the current scenario and which pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Input delays are managed on a per-scenario basis. For designs with multiple scenarios, you can specify scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command applies to the current scenario. The tool issues an error if there is no current scenario. If the **-scenarios** option is specified, the command applies to all of the specified scenarios.

If the **-modes** option is specified, the command applies to all scenarios of the specified modes. The tool issues an error if none of the specified modes have any scenarios.

If the **-corners** option is specified, the command applies to all scenarios of the specified corners and the current mode. The tool issues an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are specified, the command applies to all scenarios of the specified corners and the specified modes. The tool issues an error if there are no such scenarios. It is an error to specify **-scenarios** with **-modes** or **-corners**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

This example sets the variable inDel to the collection of input delays defined on ports IN* in the current scenario.

```
prompt> set inDel [get_input_delays -of_objects [get_ports IN*]]
```

SEE ALSO

collections(2)
set_input_delay(2)

get_instance_result

Returns a list of cell instances with rail results based on the type of result specified. The command lists the instances in the order of values from high to low.

When the -collection option is selected, this command creates a collection of cells which can be assigned to a variable or pass to another command.

SYNTAX

list **get_instance_result**
-type *result_type*
[-net *net_name*]
[-touching *region*]
[-top *top_num*]
[-threshold *threshold*]
[-percentage *percentage*]
[-histogram]
[-collection]

Data Types

result_type string
net_name string
region list
top_num integer
threshold float
percentage float

ARGUMENTS

-type *result_type*

Specifies the rail result type to report. Accepted types are effective_voltage_drop, current, effective_resistance, min_path_resistance, or power. This option is a required option.

-net *net_name*

Specifies a power or ground net name for which rail results to report. When not specified, results from all power nets will be returned.

-touching *region*

Specifies a region such that cell instances which are completely inside the specified region are returned with rail results. The region boundary can be a rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

This option is mutually exclusive with the -top, -threshold, -percentage, or -histogram option.

-top ***top_num***

Returns the top number of cell instances for the result type specified. This option is mutually exclusive with the -touching, -threshold, -percentage, or -histogram option.

-threshold ***threshold***

Returns cell instances with rail result values greater than the specified threshold. This option is mutually exclusive with the -touching, -top, -percentage, or -histogram option.

-percentage ***percentage***

Returns cell instances with rail results values such that its percentage ratio (calculated over ideal supply voltage) is greater than the specified percentage. This option is mutually exclusive with the -touching, -top, -threshold, or -histogram option.

-histogram

Returns a histogram of rail results. The default histogram has 10 bins, which are expanded from the smaller value to the biggest value. This option is mutually exclusive with the -touching, -top, -threshold, or -percentage option.

-collection

Returns a collection of cell instances with the result type. This option can be used with the -touching, -top, -threshold, or -percentage option. This option is mutually exclusive with the -histogram option.

The **get_instance_result** command returns a list of cell instances with rail result. This command has to be run after command **analyze_rail** or **open_rail_result**.

EXAMPLES

The following example returns a list of cell instances within a rectangle area:

```
prompt> get_instance_result -net VDD -type voltage_drop -touching {{1610 1968} {1620 1969}}
lsw2/lsw2_pktpro/U179070 -0.053533
lsw2/lsw2_pktpro/U64076 -0.0535949
lsw2/lsw2_pktpro/U83035 -0.0535949
lsw2/lsw2_pktpro/U181017 -0.0535949
lsw2/lsw2_pktpro/U103335 -0.05366
```

The following example returns a collection of cell instances within a rectangle area:

```
prompt> get_instance_result -net VDD -type voltage_drop -touching {{1610 1968} {1620 1969}}
{lsw2/lsw2_pktpro/U179070 lsw2/lsw2_pktpro/U64076 lsw2/lsw2_pktpro/U83035 lsw2/lsw2_pktpro/U181017 lsw2/lsw2_pktpro/U1
```

The following example returns the top 5 cell instances on the effective voltage drop:

```
prompt> get_instance_result -net VDD -type voltage_drop -top 5
lsw2/lsw2_pktpro/I12653 -0.1022
```

lsw2/latch12 -0.101123
lsw2/l123422 -0.0988
lsw2/lsw2_pktpro/U112 -0.08765
lsw2/lsw2_pktpro/U123 -0.06542

SEE ALSO

analyze_rail(2)
open_rail_result(2)

get_io_guides

Creates a collection by selecting I/O guides from the current design.

SYNTAX

collection **get_io_guides**
[-design *design_name*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design_name string
expression string
exact_count integer
minimum_count integer
patterns list
objects list
point list
region list

ARGUMENTS

-design *design_name*

Specifies the top design for finding objects. If this is not specified, objects are found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the I/O guide attributes. You can determine the I/O guide attributes by using the **list_attributes** command. If the expression evaluates to **true**, the I/O guide is

included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies the expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for I/O guides level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at** but not with **-of_objects**.

patterns

Matches the I/O guide names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-of_objects *objects*

Creates a collection containing the I/O guides of the specified pad cells. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all io_guides at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all io_guides that are completely inside the specified region and doesn't include io_guides which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all io_guides that are completely inside the specified region and the io_guides which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all io_guides which are abut/touching from inside or outside to the specified region and the io_guides whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of I/O guides by selecting I/O guides that meet the selection criteria. The command returns a collection handle if one or more I/O guides meet the selection criteria. If no I/O guides match the selection criteria, the command returns an empty string.

You can use the **get_io_guides** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of I/O guides by name pattern matching.

```
prompt> get_io_guides io_guide*
{"io_guide1", "io_guide2", "io_guide3"}
```

The following example creates a collection of io_guide with attribute filtering.

```
prompt> get_io_guides -filter {side == left}
{"io_guide1"}
```

The following example creates a collection of all I/O guides.

```
prompt> get_io_guides *
{"io_guide1", "io_guide2", "io_guide3", "groupio_guideA"}
```

The following example creates a collection of all I/O guides containing the pad cells "pad1" and "pad5".

```
prompt> get_io_guides -of_objects "pad1 pad5"
{"io_guide1", "io_guide2"}
```

SEE ALSO

add_to_io_guide(2)
collections(2)
create_io_guide(2)
filter_collection(2)
query_objects(2)
regexp(2)
remove_from_io_guide(2)
remove_io_guides(2)
report_io_guides(2)
shell.common.collection_result_display_limit(3)

get_io_rings

Creates a collection by selecting I/O rings from the current design.

SYNTAX

collection **get_io_rings**
[-design *design_name*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design_name string
expression string
exact_count integer
minimum_count integer
patterns list
of_objects list
region list
point list

ARGUMENTS

-design *design_name*

Specifies the top design for finding objects. If this is not specified, objects are found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the I/O ring attributes. You can determine the I/O ring attributes by using the **list_attributes** command. If the expression evaluates to **true**, the I/O ring is included

in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for I/O rings level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

patterns

Matches the I/O ring names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-of_objects *of_objects*

Creates a collection containing the I/O rings of the specified pad cells. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all IO rings at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all IO rings that are completely inside the specified region and doesn't include IO rings which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all IO rings that are completely inside the specified region and the IO rings which are abut/touching from inside with the specified region. The region boundary can be a rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all IO rings which are abut/touching from inside or outside to the specified region and the IO rings whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of I/O rings by selecting I/O rings that meet the selection criteria. It returns a collection handle if one or more I/O rings meet the selection criteria. If no I/O rings match the selection criteria, it returns an empty string.

You can use the **get_io_rings** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of I/O rings by name pattern matching.

```
prompt> get_io_rings io_ring*
{"io_ring1", "io_ring2", "io_ring3"}
```

The following example creates a collection of I/O ring with attribute filtering.

```
prompt> get_io_rings -filter {"outer_ring == io_ring2"}
{"io_ring1"}
```

The following example creates a collection of all I/O rings.

```
prompt> get_io_rings *
{"io_ring1", "io_ring2", "io_ring3", "groupio_ringA"}
```

The following example creates a collection of all I/O rings containing the io_rings "io_ring1.bottom" and "io_ring2.left".

```
prompt> get_io_rings -of_objects "io_ring1.bottom io_ring2.left"
{"io_ring1", "io_ring2"}
```

The following example creates a collection of all I/O rings containing the pad cells "pad1" and "pad5" (through an I/O guide associated with the I/O ring)

```
prompt> get_io_rings -of_objects "pad1 pad5"
{"io_ring1", "io_ring2"}
```

SEE ALSO

add_to_io_ring(2)
collections(2)
create_io_ring(2)
filter_collection(2)
query_objects(2)

regexp(2)
remove_from_io_ring(2)
remove_io_rings(2)
report_io_rings(2)
shell.common.collection_result_display_limit(3)

get_keepout_margins

Creates a collection of keepout margins from the current design.

SYNTAX

collection **get_keepout_margins**
[-filter *expression*]
[-quiet]
[-design *design*]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[-type hard | soft | hard_macro | routing_blockage]
[*patterns* | -of_objects *objects*]

Data Types

expression string
design string
exact_count integer
minimum_count integer
patterns list
objects list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the keepout attributes. You can determine the keepout attributes by using the **list_attributes** command. If the expression evaluates to **true**, the keepout is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Returns the keepout margins of blocks and its instances hierarchically. This option is useful only with *patterns* and not with **-of_objects**.

-type hard | **soft** | **hard_macro** | **routing_blockage**

Returns the keepout margins of specific type only. The valid values for the options are: hard, soft, hard_macro and routing_blockage

-of_objects *objects*

Creates a collection containing the keepout margins associated with the specified objects. The objects can be designs, cells and lib cells. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches the keepout names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the *pattern*.

DESCRIPTION

This command creates a collection of keepouts from the current design that match certain criteria. The command returns a collection

handle (identifier) if any keepouts match the value of the *patterns* option and pass the filter (if specified). If no objects match the criteria, the command returns an empty string.

You can use the **get_keepout_margins** command at the command prompt or nest it as an argument to another command (such as **report_keepout_margin**). You can also assign the **get_keepout_margins** result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example returns keepout margins from the current_design

```
prompt> get_keepout_margins
{lib1:design1.design/KEEPOUT_hard_INNER_0
lib1:design1.design/KEEPOUT_soft_INNER_0
lib1:design1.design/KEEPOUT_hard_macro_INNER_0
lib1:design1.design/KEEPOUT_routing_blockage_INNER_0
lib1:design1.design/KEEPOUT_hard_OUTER_0
lib1:design1.design/KEEPOUT_soft_OUTER_0
lib1:design1.design/KEEPOUT_hard_macro_OUTER_0
lib1:design1.design/KEEPOUT_routing_blockage_OUTER_0}
```

The following example returns keepout margins of type 'hard' from all cells in the design

```
prompt> get_keepout_margins -of_objects [get_cells -hierarchical] -type hard
{cntrl/KEEPOUT_hard_OUTER_0 cntrl/KEEPOUT_hard_INNER_0
cntrl/U151/KEEPOUT_hard_OUTER_3 cntrl/U121/KEEPOUT_hard_OUTER_4 ...}
```

The following example returns keepout margins from the cell 'U280'

```
prompt> get_keepout_margins -of_objects {U280}
{U280/KEEPOUT_soft_OUTER_2196}
```

The following example returns keepout margins of lib cell 'abc'

```
prompt> get_keepout_margins -of_objects [get_lib_cells */abc/frame]
{lib1:abc.frame/KEEPOUT_hard_OUTER_1968}
```

SEE ALSO

create_keepout_margin(2)
remove_keepout_margins(2)
report_keepout_margins(2)

get_label_switch_list

Gets the value of the switch list for the specified design, if set.

SYNTAX

status **get_label_switch_list**

DESCRIPTION

This command gets the label switch list of a design. A designs' label switch list is used during linking if block is the top design.

The label switch list is a precedence-ordered list of labels that is applied when binding a design. If the top design has its label switch list explicitly set, then that label switch list is used. If not, then the top design label is used.

SEE ALSO

set_label_switch_list(2)

get_latch_loop_groups

Returns a list of collections of pins. Each collection contains the transparent latch data pins for one latch loop group.

SYNTAX

list **get_latch_loop_group**
[-of_objects *pin_list*]
[-loop_breakers_only]

Data Types

pin_list list

ARGUMENTS

-of_objects *pin_list*

Specifies a collection of data pins of transparent latches. This option returns only pins of loop groups containing the specified pins. By default, the command returns one collection for each loop group in the design.

-loop_breakers_only

Specifies that the returned collection contains only pins that are loop breaker latch data pins (that have a **true** value for the **is_latch_loop_breaker** pin attribute). By default, all transparent latch data pins within the loop group are returned in the collections.

DESCRIPTION

When sequential loops of transparent latches exist in a design, a loop group containing all latch data pins of intersecting loops is formed. The **get_latch_loop_groups** command analyzes the design and returns a collection for each loop group formed. The collection contains the latch data pins of the loop group. Combinational pins are not included in the results.

You can use the results of the **get_latch_loop_groups** command to find paths within loops that violate timing. For an example, see the EXAMPLES section.

EXAMPLES

The following example reports violating paths within latch loops by using the **get_latch_loop_groups** command. No paths outside

the loops are reported.

```
prompt> foreach a_loop_group [get_latch_loop_groups -loop_breakers_only] {
          report_timing -from $a_loop_group -to $a_loop_group -slack_lesser_than 0.0
        }
```

SEE ALSO

report_latch_loop_groups(2)
set_latch_loop_breaker(2)

get_layers

Creates a collection of one or more layers.

SYNTAX

collection **get_layers**
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-all_purposes]
[*patterns* | -of_objects *of_objects*]

Data Types

expression string
exact_count int
minimum_count int
of_objects list
patterns string

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*. For any layers that match, the expression is evaluated based on the layer's attributes. If the expression evaluates to true, the layer is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-all_purposes

If this is specified, not just the tech layers, but all the layer purposes including layerDataTypes from tech file are also returned. All the layer purposes are stilll considered for a match, when the pattern contains a purpose name or purpose number. For example, if the pattern is "M1:signal" or "M1:2" or "M1:*", then all the layer purposes are also searched for a match.

-of_objects *of_objects*

Creates a collection containing the layers of the specified libs. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches layer names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark) or regular expressions, depending on whether or not you also use the **-regexp** option.

DESCRIPTION

This command creates a collection of layers defined in the technology file for the current library.

You can use the **get_layers** command at the command prompt or as an argument nested in another command (for example, in the **query_objects** command). You can also assign its result to a variable.

The layer name printed for returned collection will have following format.

layer_name:purpose_name

The same format can be used to find layers with a specific purpose. If a layer **M1** with purpose **signal** is required, following can be used to get the required layer.

```
get_layers M1:signal
```

If the purpose **signal** has the purpose number **3**, the following will also work.

```
get_layers M1:3
```

When no purpose is specified in the layer name pattern, then the layer with default purpose **drawing** will be returned. If a layer with

all its purposes is needed, then wildcard "*" can be used in the pattern name, as following.

```
get_layers M1:*
```

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a layer for the name M1R.

```
prompt> get_layers M1R
{M1R}
```

SEE ALSO

collections(2)
filter_collection(2)
query_objects(2)
shell.common.collection_result_display_limit(3)

get_lib

Creates a collection of libraries loaded into memory. You can assign these libraries to a variable or pass them into another command.

SYNTAX

collection **get_libs**
[-implicit | -explicit | -all]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *objects*]

Data Types

string *expression*
list *objects*
list *patterns*

ARGUMENTS

-implicit

Returns reference libraries implicitly opened through the ref_lib_list. Exclusive with -explicit and -all

-explicit

Returns main libraries explicitly opened with the open_lib command. Exclusive with -implicit and -all

-all

Returns both main and reference libraries. This is the default, exclusive with -implicit and -explicit.

-filter *expression*

Filters the collection with *expression*. For any libraries that match *patterns* (or *objects*), the expression is evaluated based on the library's attributes. If the expression evaluates to true, the library is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modify the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *objects*

Creates a collection of libraries that contain the specified objects. In this case, each object is either a named library cell or a library cell collection. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

patterns

Matches library names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type lib.

DESCRIPTION

The **get_libs** command creates a collection of libraries from those currently loaded into memory that match certain criteria. The command returns a collection if any libraries match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_libs** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_libs** result to a variable.

When issued from the command prompt, **get_libs** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable **collection_result_display_limit**.

The "implicit query" property of **get_libs** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_libs** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example creates a design library with two reference libraries.

```
prompt> create_lib top_lib -ref_libs {ref_lib1 ref_lib2}
{top_lib}
prompt> get_libs -all *
{top_lib ref_lib1 ref_lib2}
prompt> get_libs -explicit *
{top_lib}
prompt> get_libs -implicit *
{ref_lib1 ref_lib2}
```

The following example queries all loaded libraries. Although the output looks like a list, it is just a display.

```
prompt> get_libs -all *
{misc_cmos misc_cmos_io}
```

SEE ALSO

collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
create_lib(2)
save_lib(2)
open_lib(2)
set_ref_libs(2)
close_lib(2)
shell.common.collection_result_display_limit(3)

get_lib_cell

Creates a collection of library cells from the reference libraries loaded in memory.

SYNTAX

collection **get_lib_cells**
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hsc *separator_character*]
[-include_subcells]
[*patterns*]
[-of_objects *objects*]

Data Types

expression string
exact_count integer
minimum_count integer
separator_character string
patterns list
objects list

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any library cells that match *patterns* (or *objects*), the expression is evaluated based on the library cell's attributes. If the expression evaluates to true, the library cell is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

Regular expression matching is the same as in the Tcl **regexp** command. When using the **-regexp** option, take care in the way

you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

The **-exact** and **-regexp** options are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find.

If the command finds a different number of objects, it raises a Tcl error and stops processing.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find.

If the command finds fewer objects, it raises a Tcl error and stops processing.

-expect_each_pattern_matches

Requires that each pattern in *patterns* must match at least one object.

If one or more patterns does not match, the command raises a Tcl error and stops processing.

-hsc *separator_character*

Specifies the hierarchy separator character.

The default is /. Valid values are /, @, ^, #, ., and |.

-include_subcells

Includes child cells in the results for hierarchical library cells.

By default, the results include only the top-level library cells.

patterns

Matches library cell names against patterns.

Patterns can include the wildcard characters '*' and '?' or regular expressions, based on the **-regexp** option. Patterns can also include collections of library cells.

The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either the *patterns* argument or the **-of_objects** option, the command assumes a default pattern of '*/*'.

-of_objects *objects*

Creates a collection of library cells that are referenced by the specified cells, own the specified library cell pins, or are in the specified libraries.

The **-of_objects** option and the *patterns* option are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_lib_cells** command creates a collection of library cells that match certain criteria. The command looks for the library cells in the reference libraries currently loaded in memory. The command returns a collection if any library cells match the specified criteria. If no objects match the criteria, the command returns an empty string.

In general, the command returns only library cells with timing views. If a library does not contain any cells with timing views, the command returns library cells with frame or design views. If you specify the view type (for example '*/*/frame' or '*/OR2/frame'), the library cells with that view type are returned, if found. If a library cell does not have a timing view and the library cell name does not contain a wildcard (for example '*/FILL4'), the command returns the frame view, if it exists. Otherwise, the command returns the design view. The command returns the layout view only if you explicitly specify that view type (for example '*/FILL8/layout').

You can use **get_lib_cells** commnad with lib cell name only which will treat as */lib_cell_name

You can use the **get_lib_cells** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_lib_cells** result to a variable.

When issued from the command prompt, **get_lib_cells** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_lib_cells** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use the **get_lib_cells** command as an argument to the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example creates a collection of the library cells in the misc_cmos library whose names begin with AN2. Although the output looks like a list, it is just a display.

```
prompt> get_lib_cells misc_cmos/AN2*
{misc_cmos/AN2 misc_cmos/AN2P}
```

The following example shows one way to determine the library cell used by a particular cell instance.

```
prompt> get_lib_cells -of_objects [get_cells o_reg1]
{misc_cmos/FD2}
```

SEE ALSO

collections(2)
filter_collection(2)
get_cells(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_lib_cells

Creates a collection of library cells from the reference libraries loaded in memory.

SYNTAX

collection **get_lib_cells**
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hsc *separator_character*]
[-include_subcells]
[*patterns*]
[-of_objects *objects*]

Data Types

expression string
exact_count integer
minimum_count integer
separator_character string
patterns list
objects list

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any library cells that match *patterns* (or *objects*), the expression is evaluated based on the library cell's attributes. If the expression evaluates to true, the library cell is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

Regular expression matching is the same as in the Tcl **regexp** command. When using the **-regexp** option, take care in the way

you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

The **-exact** and **-regexp** options are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find.

If the command finds a different number of objects, it raises a Tcl error and stops processing.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find.

If the command finds fewer objects, it raises a Tcl error and stops processing.

-expect_each_pattern_matches

Requires that each pattern in *patterns* must match at least one object.

If one or more patterns does not match, the command raises a Tcl error and stops processing.

-hsc *separator_character*

Specifies the hierarchy separator character.

The default is /. Valid values are /, @, ^, #, ., and |.

-include_subcells

Includes child cells in the results for hierarchical library cells.

By default, the results include only the top-level library cells.

patterns

Matches library cell names against patterns.

Patterns can include the wildcard characters '*' and '?' or regular expressions, based on the **-regexp** option. Patterns can also include collections of library cells.

The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either the *patterns* argument or the **-of_objects** option, the command assumes a default pattern of '*/*'.

-of_objects *objects*

Creates a collection of library cells that are referenced by the specified cells, own the specified library cell pins, or are in the specified libraries.

The **-of_objects** option and the *patterns* option are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_lib_cells** command creates a collection of library cells that match certain criteria. The command looks for the library cells in the reference libraries currently loaded in memory. The command returns a collection if any library cells match the specified criteria. If no objects match the criteria, the command returns an empty string.

In general, the command returns only library cells with timing views. If a library does not contain any cells with timing views, the command returns library cells with frame or design views. If you specify the view type (for example '*/*/frame' or '*/OR2/frame'), the library cells with that view type are returned, if found. If a library cell does not have a timing view and the library cell name does not contain a wildcard (for example '*/FILL4'), the command returns the frame view, if it exists. Otherwise, the command returns the design view. The command returns the layout view only if you explicitly specify that view type (for example '*/FILL8/layout').

You can use **get_lib_cells** commnad with lib cell name only which will treat as */lib_cell_name

You can use the **get_lib_cells** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_lib_cells** result to a variable.

When issued from the command prompt, **get_lib_cells** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_lib_cells** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use the **get_lib_cells** command as an argument to the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example creates a collection of the library cells in the misc_cmos library whose names begin with AN2. Although the output looks like a list, it is just a display.

```
prompt> get_lib_cells misc_cmos/AN2*
{misc_cmos/AN2 misc_cmos/AN2P}
```

The following example shows one way to determine the library cell used by a particular cell instance.

```
prompt> get_lib_cells -of_objects [get_cells o_reg1]
{misc_cmos/FD2}
```

SEE ALSO

collections(2)
filter_collection(2)
get_cells(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_lib_pin

Creates a collection of library cell pins from libraries loaded into memory.

SYNTAX

collection **get_lib_pins**
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hsc *separator_character*]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*
| -all]

Data Types

expression string
exact_count integer
minimum_count integer
separator_character string
patterns list
objects collection
point list
region list

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any library cell pins that match *patterns* (or *objects*), the expression is evaluated based on the library cell pin's attributes. If the expression evaluates to true, the library cell pin is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

The **-exact** and **-regexp** options are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Specifies that each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hsc separator_character

Specifies a separator character. Valid values are /, @, ^, #, ., |.

The default is /.

-of_objects *objects*

Creates a collection of library cell pins referenced by the specified netlist pins or owned by the specified library cells. In this case, each object is either a named library cell, a named netlist pin, a library cell collection, or a netlist pin collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

patterns

Creates a collection that contains all library cell pins that match the specified patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

-at *point*

Creates a collection that contains all library cell pins at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one

except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

-within *region*

Creates a collection that contains all library cell pins that are completely inside the specified region, not including library cell pins that abut the specified region from the inside. The region boundary can be a rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

-touching *region*

Creates a collection that contains all library cell pins that are either completely inside the specified or abut the specified region from the inside. the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

-intersect *region*

Creates a collection that contains all library cell pins that abut the specified region from the inside or outside and the library cell pins that are partially outside of the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

-all

Creates a collection that contains all library cell pins in the libraries loaded in memory. This is the same as using "*/*/*" as the *pattern*.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

DESCRIPTION

The **get_lib_pins** command creates a collection of library cell pins from the libraries loaded into memory that match certain criteria. The command returns a collection if any library cell pins match the *patterns* or *objects* arguments and pass the filter (if specified). If no objects matched the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_lib_pins** command at the command prompt, or you can nest it as an argument to another command such as the **query_objects** command. In addition, you can assign the **get_lib_pins** result to a variable.

When issued from the command prompt, the **get_lib_pins** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, the command displays a maximum of 100 objects. To change this maximum, set the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of the **get_lib_pins** command provides a fast, simple way to display the pins in the collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use the **get_lib_pins** command as an argument to the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries all pins of the AN2 library cell in the misc_cmos library. Although the output looks like a list, it is just a display.

```
prompt> [get_lib_pins misc_cmos/AN2/*
{misc_cmos/AN2/A misc_cmos/AN2/B misc_cmos/AN2/Z}
```

The following example shows one way to find out how the library pin is used by a particular pin in the netlist.

```
prompt> get_lib_pins -of_objects o_reg1/Q
{misc_cmos/FD2/Q}
```

SEE ALSO

collections(2)
filter_collection(2)
get_libs(2)
get_lib_cells(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_lib_pins

Creates a collection of library cell pins from libraries loaded into memory.

SYNTAX

collection **get_lib_pins**
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hsc *separator_character*]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*
| -all]

Data Types

expression string
exact_count integer
minimum_count integer
separator_character string
patterns list
objects collection
point list
region list

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any library cell pins that match *patterns* (or *objects*), the expression is evaluated based on the library cell pin's attributes. If the expression evaluates to true, the library cell pin is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

The **-exact** and **-regexp** options are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Specifies that each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hsc separator_character

Specifies a separator character. Valid values are /, @, ^, #, ., |.

The default is /.

-of_objects *objects*

Creates a collection of library cell pins referenced by the specified netlist pins or owned by the specified library cells. In this case, each object is either a named library cell, a named netlist pin, a library cell collection, or a netlist pin collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

patterns

Creates a collection that contains all library cell pins that match the specified patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

-at *point*

Creates a collection that contains all library cell pins at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one

except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

-within *region*

Creates a collection that contains all library cell pins that are completely inside the specified region, not including library cell pins that abut the specified region from the inside. The region boundary can be a rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

-touching *region*

Creates a collection that contains all library cell pins that are either completely inside the specified or abut the specified region from the inside. the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

-intersect *region*

Creates a collection that contains all library cell pins that abut the specified region from the inside or outside and the library cell pins that are partially outside of the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

-all

Creates a collection that contains all library cell pins in the libraries loaded in memory. This is the same as using "*/*/*" as the *pattern*.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, **-at**, and **-all** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses "*/*/*" as the *pattern*.

DESCRIPTION

The **get_lib_pins** command creates a collection of library cell pins from the libraries loaded into memory that match certain criteria. The command returns a collection if any library cell pins match the *patterns* or *objects* arguments and pass the filter (if specified). If no objects matched the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_lib_pins** command at the command prompt, or you can nest it as an argument to another command such as the **query_objects** command. In addition, you can assign the **get_lib_pins** result to a variable.

When issued from the command prompt, the **get_lib_pins** command behaves as though the **query_objects** command had been called to display the objects in the collection. By default, the command displays a maximum of 100 objects. To change this maximum, set the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of the **get_lib_pins** command provides a fast, simple way to display the pins in the collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use the **get_lib_pins** command as an argument to the **query_objects** command.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries all pins of the AN2 library cell in the misc_cmos library. Although the output looks like a list, it is just a display.

```
prompt> [get_lib_pins misc_cmos/AN2/*
{misc_cmos/AN2/A misc_cmos/AN2/B misc_cmos/AN2/Z}
```

The following example shows one way to find out how the library pin is used by a particular pin in the netlist.

```
prompt> get_lib_pins -of_objects o_reg1/Q
{misc_cmos/FD2/Q}
```

SEE ALSO

collections(2)
filter_collection(2)
get_libs(2)
get_lib_cells(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_lib_timing_arcs

Creates a collection of library arcs for custom reporting and other processing. You can assign these library arcs to a variable and get the desired attribute for further processing.

SYNTAX

string **get_lib_timing_arcs**
[-from *from_list*]
[-to *to_list*]
[-of_objects *object_list*]
[-filter *expression*]
[-quiet]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]

Data Types

from_list list
to_list list
of_objects list
expression string
exact_count int
minimum_count int

ARGUMENTS

-to *to_list*

Specifies the "to" library pins, or ports. All backward library arcs from the specified library pins or ports are considered.

-from *from_list*

Specifies the "from" library pins, or ports. All forward library arcs from the specified library pins or ports are considered.

-of_objects *object_list*

Specifies library cells or timing arcs. If a library cell is specified, all library cell arcs of that cell are considered. If a timing arc collection is given in the object list, the corresponding library timing arc is considered.

-filter *expression*

Specifies the filter expression. A filter expression is a string that comprises a series of logical expressions describing a set of constraints you want to place on the collection of library arcs. Each subexpression of a filter expression is a relation contrasting an attribute name with a value, by means of an operator.

-quiet

Specifies that all messages are to be suppressed.

-expect ***exact_count***

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least ***minimum_count***

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

DESCRIPTION

Creates a collection of library arcs for custom reporting and other processing. You can assign these library arcs to a variable and get the desired attribute for further processing. Use the **foreach_in_collection** command to iterate among the library arcs in the collection. You can use the **get_attribute** command to obtain information about the paths. You can also use the filter expression to filter the library arcs that satisfy the specified conditions. The following attributes are supported on library timing arcs:

from_lib_pin
is_disabled
is_user_disabled
mode
object_class
sdf_cond
sense
to_lib_pin

One attribute of a library timing arc is the **from_lib_pin**, which is the library pin from which the timing arc begins. In the same way, **to_lib_pin** is the library pin to which the timing arc ends. To get more information on **from_lib_pin** and **to_lib_pin**, use the **get_attributes** command. The **object_class** attribute holds the class name of library timing arcs: **lib_timing_arc**.

See the **collections** and **foreach_in_collection** man pages for more information.

Multicorner-Multimode Support

EXAMPLES

The following procedure is an example of how the collection returned from an invocation of **get_lib_timing_arcs** can be utilized to provide useful and readable cell level arc information.

```
proc show_lib_arcs {args} {
  set lib_arcs [eval [concat get_lib_timing_arcs $args]]
  echo [format "%15s    %-15s  %18s" "from_lib_pin" "to_lib_pin" \
        "sense"]
  echo [format "%15s    %-15s  %18s" "------------" "----------" \
        "-----"]

  foreach_in_collection lib_arc $lib_arcs {
    set fpin [get_attribute $lib_arc from_lib_pin]
    set tpin [get_attribute $lib_arc to_lib_pin]
```

```
    set sense [get_attribute $lib_arc sense]
    set from_lib_pin_name [get_attribute $fpin base_name]
    set to_lib_pin_name [get_attribute $tpin base_name]
    echo [format "%15s -> %-15s  %18s" \
       $from_lib_pin_name $to_lib_pin_name \
       $sense]
  }
}
```

prompt> **show_lib_arc -of_objects [get_timing_arcs -of_objects ffa]**

```
  from_lib_pin    to_lib_pin                sense
  ------------    ----------                -----
          CP -> D                setup_clk_rise
          CP -> D                 hold_clk_rise
          CP -> Q                   rising_edge
          CP -> QN                  rising_edge
          CD -> Q                   clear_low
          CD -> QN                   preset_low
```

prompt> **show_lib_arc -of_objects mylib/AN2**

```
  from_lib_pin    to_lib_pin                sense
  ------------    ----------                -----
          A -> Z                positive_unate
          B -> Z                positive_unate
```

prompt> **show_lib_arc -from mylib/AN2/A**

```
  from_lib_pin    to_lib_pin                sense
  ------------    ----------                -----
          A -> Z                positive_unate
```

SEE ALSO

collections(2)
filter_collection(2)
foreach_in_collection(2)
get_attribute(2)
get_timing_arcs(2)

get_libs

Creates a collection of libraries loaded into memory. You can assign these libraries to a variable or pass them into another command.

SYNTAX

collection **get_libs**
[-implicit | -explicit | -all]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *objects*]

Data Types

string *expression*
list *objects*
list *patterns*

ARGUMENTS

-implicit

Returns reference libraries implicitly opened through the ref_lib_list. Exclusive with -explicit and -all

-explicit

Returns main libraries explicitly opened with the open_lib command. Exclusive with -implicit and -all

-all

Returns both main and reference libraries. This is the default, exclusive with -implicit and -explicit.

-filter *expression*

Filters the collection with *expression*. For any libraries that match *patterns* (or *objects*), the expression is evaluated based on the library's attributes. If the expression evaluates to true, the library is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modify the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *objects*

Creates a collection of libraries that contain the specified objects. In this case, each object is either a named library cell or a library cell collection. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

patterns

Matches library names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type lib.

DESCRIPTION

The **get_libs** command creates a collection of libraries from those currently loaded into memory that match certain criteria. The command returns a collection if any libraries match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_libs** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_libs** result to a variable.

When issued from the command prompt, **get_libs** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable **collection_result_display_limit**.

The "implicit query" property of **get_libs** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_libs** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example creates a design library with two reference libraries.

```
prompt> create_lib top_lib -ref_libs {ref_lib1 ref_lib2}
{top_lib}
prompt> get_libs -all *
{top_lib ref_lib1 ref_lib2}
prompt> get_libs -explicit *
{top_lib}
prompt> get_libs -implicit *
{ref_lib1 ref_lib2}
```

The following example queries all loaded libraries. Although the output looks like a list, it is just a display.

```
prompt> get_libs -all *
{misc_cmos misc_cmos_io}
```

SEE ALSO

collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
create_lib(2)
save_lib(2)
open_lib(2)
set_ref_libs(2)
close_lib(2)
shell.common.collection_result_display_limit(3)

get_license

Obtains a license feature.

SYNTAX

status **get_licenses**
feature_name
[-quantity *num_licenses*]

Data Types

feature_name	string
num_licenses	integer

ARGUMENTS

feature_name

Specifies the feature name to be obtained.

-quantity *num_licenses*

Specifies the total number of licenses to be checked out for the feature. If some licenses have already been checked out, the command acquires only the additional licenses needed to bring the total to the specfied quantity. If this option is not specified, only one license is checked out for the feature.

DESCRIPTION

This command checks out a license for the specified feature. The features remain checked out until the **remove_licenses** command is used or until the program exits.

If multiple licenses are required, you can use the **-quantity** option to specify the total number of licenses needed. You can use this option to reserve the required number of licenses early in the session, rather than risking a failure later in the session if the licenses are not available.

The **list_licenses** command provides a list of the features that you are currently using.

EXAMPLES

The following example obtains the requested feature:

```
prompt> get_licenses Fusion-Compiler-BE
1
```

The following example obtains two licenses of the feature:

```
prompt> get_licenses Fusion-Compiler-BE -quantity 2
1
```

SEE ALSO

check_license(2)
list_licenses(2)
remove_licenses(2)

get_licenses

Obtains a license feature.

SYNTAX

status **get_licenses**
feature_name
[-quantity *num_licenses*]

Data Types

feature_name	string
num_licenses	integer

ARGUMENTS

feature_name

Specifies the feature name to be obtained.

-quantity *num_licenses*

Specifies the total number of licenses to be checked out for the feature. If some licenses have already been checked out, the command acquires only the additional licenses needed to bring the total to the specfied quantity. If this option is not specified, only one license is checked out for the feature.

DESCRIPTION

This command checks out a license for the specified feature. The features remain checked out until the **remove_licenses** command is used or until the program exits.

If multiple licenses are required, you can use the **-quantity** option to specify the total number of licenses needed. You can use this option to reserve the required number of licenses early in the session, rather than risking a failure later in the session if the licenses are not available.

The **list_licenses** command provides a list of the features that you are currently using.

EXAMPLES

The following example obtains the requested feature:

```
prompt> get_licenses Fusion-Compiler-BE
1
```

The following example obtains two licenses of the feature:

```
prompt> get_licenses Fusion-Compiler-BE -quantity 2
1
```

SEE ALSO

check_license(2)
list_licenses(2)
remove_licenses(2)

get_macro_group_packing_clone_candidates

Get macro group packing canditates.

SYNTAX

collection_of_objects **get_macro_group_packing_clone_candidates**
-group macro_group

ARGUMENTS

-group macro_group

Specifies source macro group.

DESCRIPTION

This command returns a collection of macro groups which the packing of source macro group can be copied to. The command goes over every macro group in the ndm database, and determine whether the source macro group can be copied to it using following steps.

\n+[step]. If the number of macros of source macro group and destination macro group are not the same, it will not copy the packing from source to desitination and the skip the group.
\n+[step]. Sorting macros of source macro group and destinaiton macro group based on macro names. Each macro in the source macro list corresponds to the macro of destinaiton macro list at the same location in the list.
\n+[step]. Comparing corresponding macros from these two macro list one by one. If any two macros have different size, the packing cannot be copied and the group is skipped.
\n+[step]. Add the macro group to the result.

EXAMPLES

The following command returns a list of macro groups which can be copied from macro group G1.

prompt> **get_macro_group_packing_clone_candidatess** -group G1

SEE ALSO

create_macro_group(2)
clone_macro_group_packing(2)

get_matching_types

Creates a collection by selecting matching types from the current block.

SYNTAX

collection **get_matching_types**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *of_objects*]

Data Types

design collection
expression string
exact_count int
minimum_count int
of_objects list
patterns list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the matching type attributes. You can determine the matching type attributes by using the **list_attributes** command. If the expression evaluates to **true**, the matching type is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *of_objects*

Creates a collection containing the matching types of the specified cells, pins, or terminals. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches the matching type names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

DESCRIPTION

This command creates a collection of matching types by selecting matching types that meet the selection criteria. It returns a collection handle if one or more matching types meet the selection criteria. If no matching types match the selection criteria, it returns an empty string.

You can use the **get_matching_types** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of matching types by name pattern matching.

```
prompt> get_matching_types myMatchingType*
{"myMatchingType1", "myMatchingType2", "myMatchingType3"}
```

The following example creates a collection of matching types with attribute filtering.

```
prompt> get_matching_types -filter {uniquify_number == 3}
{"myMatchingType2"}
```

The following example creates a collection of all matching types.

```
prompt> get_matching_types *
{"myMatchingType1", "myMatchingType2", "myMatchingType3", "yourMatchingType1"}
```

The following example creates a collection of all matching_types containing the cells "inv1".

```
prompt> get_matching_types -of_objects [get_cells inv1]
{"myMatchingType3"}
```

SEE ALSO

add_to_matching_type(2)
collections(2)
create_matching_type(2)
filter_collection(2)
query_objects(2)
regexp(2)
remove_from_matching_type(2)
remove_matching_types(2)
report_matching_types(2)
shell.common.collection_result_display_limit(3)

get_mem

Retrieves the total memory allocated by the current process.

SYNTAX

int **get_mem**

DESCRIPTION

The *get_mem* command returns the size of memory currently allocated by the current process for the purposes of storing design netlist, design annotations, and constraints information as well as computed timing information. The value printed represents the amount in kilobytes (kB).

Note that the value reported would not match values obtained by the user through Unix process tracking commands, such as *top*. This is the case because the *get_mem* command does not track memory used to load the executable and maintain process stack space. Also, it does not differentiate between resident and swapped memory.

```
prompt> get_mem
8092
```

SEE ALSO

get_cputime(2)

get_message_ids

Get application message ids

SYNTAX

string **get_message_ids** [-type *severity*]
[*pattern*]

string *severity*
string *pattern*

ARGUMENTS

-type *severity*

Filter ids based on type (Values: Info, Warning, Error, Severe, Fatal)

pattern

Get IDs matching pattern (default: *)

DESCRIPTION

The **get_message_ids** command retrieves the error, warning and informational messages used by the application. The result of this command is a Tcl formatted list of all message ids. Information about the id can be queried with the **get_message_info** command.

EXAMPLES

The following code finds all error messages and makes the application stop script execution when one of these messages is encountered.

```
foreach id [get_message_ids -type Error] {
  set_message_info -stop_on -id $id
}
```

SEE ALSO

print_message_info(2)
set_message_info(2)
suppress_message(2)

get_message_info

Returns information about diagnostic messages.

SYNTAX

integer **get_message_info**
[-error_count | -warning_count | -info_count
| -limit l_id | -occurrences o_id | -suppressed s_id | -id i_id]

Data Types

l_id string
o_id string
s_id string
i_id string

ARGUMENTS

-error_count

Returns the number of error messages issued so far.

-warning_count

Returns the number of warning messages issued so far.

-info_count

Returns the number of informational messages issued so far.

-limit *l_id*

Returns the current user-specified limit for a given message ID. The limit was set with the **set_message_info** command.

-occurrences *o_id*

Returns the number of occurrences of a given message ID.

-suppressed *s_id*

Returns the number of times a message was suppressed either using **suppress_message** or due to exceeding a user-specified limit.

-id *i_id*

Returns information about the specified message. The information is returned as a Tcl list compatible with the array set command.

DESCRIPTION

The **get_message_info** command retrieves information about error, warning, and informational messages. For example, if the following message is generated, information about it is recorded:

```
Error: unknown command 'wrong_command' (CMD-005)
```

It is useful to be able to retrieve recorded information about generated diagnostic messages. For example, you can stop a script after a certain number of errors have occurred, or monitor the number of messages issued by a single command.

You can also find out how many times a specific message has occurred, or how many times it has been suppressed. Also, you can find out if a limit has been set for a particular message ID.

EXAMPLES

The following example uses the **get_message_info** command to count the number of errors that occurred during execution of a specific command, and to return from the procedure if the error count exceeds a given amount:

```
prompt> proc \
  do_command {limit} {
    set current_errors [get_message_info -error_count]
    command
    set new_errors [get_message_info -error_count]
    if {[expr $new_errors - $current_errors] > $limit} {
      return -code error "Too many errors"
    }
    ...
  }
```

The following example uses the **get_message_info** command to retrieve information on the CMD-014 message:

```
prompt> get_message_info -id CMD-014
id CMD-014 severity Error limit 0 occurrences 0 suppressed 0 message
{Invalid %s value '%s' in list.}
```

SEE ALSO

print_message_info(2)
set_message_info(2)
suppress_message(2)
get_message_ids(2)

get_mib_objects

Creates a collection of multiply instantiated blocks (MIBs) from the current design relative to the current instance. You can assign these objects to a variable or pass them into another command.

SYNTAX

collection **get_mib_objects**
[-quiet]
[-add]
object_list

Data Types

object_list list

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-add

Include original selected pins and/or cells in returned MIB objects.

object_list

A collection or selection set of objects to get MIB from.

DESCRIPTION

The **get_mib_objects** command creates a collection of multiply instantiated blocks (MIBs) from the current design relative to the current instance.

You can use the **get_mib_objects** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_mib_objects** result to a variable.

EXAMPLES

The following example get the MIB objects for the cell named block_0.

```
prompt> get_mib_objects [get_cell block_0]
{"block_0" "block_1"}
```

The following example gets the MIB objects of a specified pin.

```
prompt> set pinsel [get_pin o_reg1/CP]
{"o_reg1/CP"}
prompt> get_mib_objects $pinsel
{"o_reg1/CP", "o_reg2/CP"}
```

SEE ALSO

check_mib_alignment(2)
check_mib_for_pin_placement(2)
collections(2)
filter_collection(2)
get_pins(2)
link_block(2)
query_objects(2)
report_mibs(2)

get_mismatch_objects

Create a collection of mismatch objects from the current design and its ref_libs. This collection does not contain mismatch objects of **accepted** mismatch state. To obtain **accepted** mismatch objects, use **repair_status** option. You can assign these objects to a variable or pass them into another command.

SYNTAX

collection **get_mismatch_objects**
[-filter *expression*]
[-repair_status *status*
[-mismatch_type *type*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
patterns | -of_objects *objects*

Data Types

expression string
status string
type string
exact_count int
minimum_count int
patterns list
of_objects list

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any cells that match *patterns* the expression is evaluated based on the mismatch object's attributes. If the expression evaluates to **true**, the mismatch object is included in the result.

-repair_status

Specifies the list of mismatch states to return mismatches. Valid values are **repaired**, **not_repaired**, **accepted**, **deleted**, and **ignored**.

-mismatch_type *type*

Specifies the list of mismatch types to return mismatches. Valid values are **bus_blast_type**, **lib_missing_logical_port**,

lib_missing_physical_port, **lib_missing_physical_reference**, **lib_name_case**, **lib_port_direction**, **lib_port_name_synonym**, **lib_port_type**, **missing_logical_reference**, **missing_port**, **port_name_case**, **cell_name_case**, **port_name_synonym**, and **empty_logic_module**.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *objects*

Creates a collection of mismatch objects associated to the specified object. In this case, each object in the list is a name, pattern, or collection. The objects can be design, lib_cell and libs. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

patterns

Matches mismatch object names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. By default, the command uses * (asterisk) as the *pattern*.

DESCRIPTION

The get_mismatch_objects command creates a collection of mismatch objects that match certain criteria. The command returns a collection handle (identifier) if any mismatch object match the value of the *patterns* option and pass the filter (if specified). If no objects match the criteria, the command returns Error.

By default, this command returns mismatch objects having any mismatch state except **accepted**. To get **accepted** mismatch objects, use **repair_status** option.

When **mismatch_type** option is used, this command will return mismatches for specified mismatch type.

You can use the **get_mismatch_objects** command at the command prompt or nest it as an argument to another command . You can also assign the **get_mismatch_objects** result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example returns all mismatch objects from current design and ref libs.

```
prompt> get_mismatch_objects
{port_name_case_ORCA_inv0d1/i port_name_case_ORCA_invbd2/i}
```

The following example returns all mismatch objects from all designs.

```
prompt> get_mismatch_objects -of_objects [get_designs]
{port_name_case_REG_FILE_inv0d1/i port_name_case_ORCA_inv0d1/i port_name_case_ORCA_invbd2/i}
```

The following example returns mismatch objects having repair state as "accepted" from current design and ref libs.

```
prompt> get_mismatch_objects -repair_status accepted
{port_name_case_REG_FILE_inv0d2/i}
```

The following example returns mismatch objects of specified mismatch states from current design and ref libs.

```
prompt> get_mismatch_objects -repair_status {accepted repaired}
{port_name_case_REG_FILE_inv0d2/i}
```

The following example returns mismatch objects of specified mismatch types from current design and ref libs.

```
prompt> get_mismatch_objects -mismatch_type port_name_case
{port_name_case_REG_FILE_inv0d2/i}
```

SEE ALSO

get_mismatch_types(2)
create_mismatch_config(2)
set_current_mismatch_config(2)
get_current_mismatch_config(2)
report_design_mismatch(2)
report_mismatch_configs(2)

get_mismatch_types

Create a collection mismatch types from the current design. You can assign these to a variable or pass them into another command.

SYNTAX

collection **get_mismatch_types**
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
patterns

Data Types

expression string
exact_count int
minimum_count int
patterns list

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any cells that match *patterns* the expression is evaluated based on the mismatch object's attributes. If the expression evaluates to **true**, the mismatch object is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect ***exact_count***

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least ***minimum_count***

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches mismatch object names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. By default, the command uses * (asterisk) as the *pattern*.

DESCRIPTION

The get_mismatch_types command creates a collection of mismatch types that match certain criteria. The command returns a collection handle (identifier) if any mismatch types match the value of the *patterns* option and pass the filter (if specified). If no objects match the criteria, the command returns Error.

You can use the **get_mismatch_types** command at the command prompt or nest it as an argument to another command . You can also assign the **get_mismatch_types** result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example returns all mismatch types.

```
prompt> get_mismatch_types
{lib_missing_physical_reference lib_missing_logical_port
lib_missing_physical_port lib_port_type lib_port_direction
lib_port_name_synonym missing_logical_reference reference_name_case
missing_port port_name_case port_name_synonym bus_bit_naming
lib_cell_name_case lib_port_name_case lib_missing_rail_pg
layer_missing_prefer_direction lib_port_missing_via_region
lib_port_missing_access_edge macro_cell_contain_too_many_obs
exceed_ndr_limitation empty_logic_module}
```

The following example queries the mismatch types which start with "lib".

```
prompt> get_mismatch_types "lib*"
{lib_missing_physical_reference lib_missing_logical_port
```

```
lib_missing_physical_port lib_port_type lib_port_direction
lib_port_name_synonym lib_cell_name_case lib_port_name_case
lib_missing_rail_pg lib_port_missing_via_region lib_port_missing_access_edge}
```

The following example returns the current repair subscript attribute for the specified mismatch type.

```
prompt> get_attr [get_mismatch_types lib_missing_logical_port] \
  current_repair
auto_fix create_placeholder_logic_lib_port default null
```

The following example returns the current repair subscript attribute for the specified configuration and mismatch type.

```
prompt> get_attr [get_mismatch_types lib_missing_logical_port] \
  current_repair(auto_fix)
create_placeholder_logic_lib_port
```

SEE ALSO

get_mismatch_objects(2)
create_mismatch_config(2)
set_current_mismatch_config(2)
get_current_mismatch_config(2)
report_design_mismatch(2)
report_mismatch_configs(2)

get_modes

Creates a collection of modes in the current design. You can assign these modes to a variable or pass them into another command.

SYNTAX

collection **get_modes**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns*]

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with *expression*. For any objects that match, the expression is evaluated based on the object's attributes. If the expression evaluates to true, the object is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches mode names against patterns. Patterns can include the wildcard characters "*" and "?".

DESCRIPTION

The **get_modes** command creates a collection of modes matching *patterns* in the current design and which pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Multicorner-Multimode Support

This command works on all modes.

EXAMPLES

This script performs **report_clock** one time for each mode beginning with "Test".

```
foreach_in_collection mode [get_modes Test*] {
 current_mode $mode
 report_clock
}
```

SEE ALSO

all_modes(2)

collections(2)
create_mode(2)
current_mode(2)
report_clocks(2)

get_modules

Creates a collection of one or more modules in a block. You can assign these modules to a variable or pass them into another command.

SYNTAX

collection **get_modules**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
patterns

Data Types

design string or collection
expression string
exact_count int
minimum_count int
patterns list

ARGUMENTS

-design

Specifies the block for finding objects. If this is not specified, objects will be found in the current block.

-filter *expression*

Filters the collection with *expression*. For any modules that match *patterns*, the expression is evaluated based on the design's attributes. If the expression evaluates to true, the design is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the

=~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches design names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type design.

DESCRIPTION

The **get_modules** command creates a collection of modules from the specified block or the current block that match certain criteria. The command returns a collection if any modules match the *patterns* and pass the filter (if specified). If no objects matched your criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_modules** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_modules** result to a variable.

When issued from the command prompt, **get_modules** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_modules** provides a fast, simple way to display modules in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_modules** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the modules that begin with 'mpu.'

```
prompt> get_modules mpu*
{"mpu_0_0", "mpu_0_1", "mpu_1_0", "mpu_1_1"}
```

SEE ALSO

collections(2)
filter_collection(2)
get_blocks(2)
get_designs(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_msg

Gets message properties.

SYNTAX

integer **get_msg**
[*msgTag*]
[-level]
[-default_level]
[-print_count]
[-trigger_count]
[-limit]
[-format]
[-last]
[-total_print_count off | debug | verbose | info | warning | error | fatal]

Data Types

msgTag string

ARGUMENTS

msgTag

Specifies the unique message tag to query.

-level

Returns the current level of the message.

-default_level

Returns the system default level of the message.

-print_count

Returns the number of times the message was printed.

-trigger_count

Returns the number of times the messages was encountered. Messages can be triggered but not printed due to their level.

-limit

Returns the print limit of the message.

-format

Returns the message format string.

-last

Returns the output of the last time the message was printed.

-total_print_count off | debug | verbose | info | warning | error | fatal

Returns the total amount of messages printed for the specified message level. Valid values are off, debug, verbose, info, warning, error, or fatal.

DESCRIPTION

The following example returns the message level for message ABC-1234.

EXAMPLES

```
prompt> get_msg ABC-1234 -level
Warning
```

The following example returns how often the specified message was printed.

```
prompt> get_msg ABC-1234 -print_count
2
```

The following example returns how often the message was triggered.

```
prompt> get_msg ABC-1234 -trigger_count
5034
```

SEE ALSO

report_msg(2)
set_msg(2)

get_multisource_clock_sink_groups

Returns a collection of sink groups defined for multisource clock tap assignment

SYNTAX

collection **get_multisource_clock_sink_groups**
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-filter *expression*]
patterns | -of_objects *objects*

Data Types

exact_count integer
minimum_count integer
expression string
patterns string
objects collection

ARGUMENTS

-quiet

Suppresses warning and error messages if no sink groups match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for sink groups that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of sink groups to find. If the command finds a different number of sink groups, a Tcl error will be raised and command processing will stop.

-expect_at_least ***minimum_count***

Specifies a minimum number of sink groups to find. If the command finds fewer sink groups, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one sink group. If one or more patterns do not match, a Tcl error will be raised and command processing will stop.

-filter ***expression***

Filters the collection with *expression*. For any sink group that match *patterns* (or *objects*) the expression is evaluated based on the sink group's attributes. If the expression evaluates to true, the sink group is included in the result.

patterns

Matches sink group names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type multisource_sink_group. *patterns* and **-of_objects** are mutually exclusive; you must specify one, but not both.

-of_objects ***objects***

Creates a collection of sink groups associated to the specified objects. In this case, each object in the list is a name, pattern, or collection. The objects can be pin or ports and refer to the driver object or collection of sinks. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_multisource_clock_sink_groups** command is used to return a collection of sink groups defined for multisource clock tap assignment. The command returns a collection if any sink groups match the *patterns* or *objects* and pass the filter (if specified). If no sink group matches the criteria, the empty string is returned.

Regular expression matching using option *-regexp* is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This example returns a sink groups that start with e prefix my_sinks.

prompt> **get_multisource_clock_sink_groups my_sinks***

This example returns the sink group that is associated with pin reg1/CK.

prompt> **get_multisource_clock_sink_groups -of_objects [get_pin reg1/CK]**

SEE ALSO

add_to_multisource_clock_sink_group(2)
create_multisource_clock_sink_group(2)
remove_from_multisource_clock_sink_group(2)
remove_multisource_clock_sink_groups(2)
report_multisource_clock_sink_groups(2)
set_multisource_clock_tap_options(2)
synthesize_multisource_clock_taps(2)

get_net

Creates a collection of nets from the netlist. You can assign these nets to a variable or pass them to another command.

SYNTAX

collection **get_nets**
[-design *design*]
[-hierarchical]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-physical_context]
[-top_net_of_hierarchical_group]
[-segments]
[-boundary_type lower | upper | both]
[-hsc *separator*]
[-include_shielded]
[-shielded_only]]
patterns | -of_objects *objects*

Data Types

design string
expression string
exact_count integer
minimum_count integer
separator character
patterns list
objects list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this option is not specified, objects will be found in the current design.

-hierarchical

Searches for nets within the hierarchy relative to the current instance, when used without the **-physical_context** option. The full

name of the object at a particular level of hierarchy must match *patterns*. The search is similar to the UNIX **find** command. For example, if there is a net u1/muxsel, a hierarchical search would find it using "muxsel". **-hierarchical** and **-of_objects**, can be used together only when **-physical_context** is also used and the *objects* argument to **-of_objects** are nets. When used together with **-physical_context** option, the search is performed level-by-level, where each level is a physical hierarchy. So, if u1 is a logical hierarchical instance, then the search will **not** find the net "muxsel". However, search for "*muxsel", would find the net physical net u1/muxsel.

-filter *expression*

Filters the collection with *expression*. For any nets that match *patterns* (or *objects*), the expression is evaluated based on the attributes of the cell. If the expression evaluates to true, the net is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Considers the *patterns* argument as true regular expressions rather than a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-physical_context

Searches only for physical nets. This option can also be used with **-of_objects**. If the *objects* argument to **-of_objects** is a pin, the physical net connected to the pin is returned. If the *objects* argument to **-of_objects** is a cell, a collection of physical nets connected to the pins of the cell is returned. If *objects* argument to **-of_objects** is a net, the physical net corresponding to the net is returned. If **-of_objects** is not used, the physical nets matching the specified *patterns* are returned. When **-hierarchical** is used, the patterns are matched against the name of the physical nets, which typically correspond to full hierarchical name of logical nets. Therefore, a search for "net*" will only find nets in the topmost logical hierarchy, matching the pattern "net*". To find physical nets in lower levels of the logical hierarchy, prefix the expression with an asterisk (*), for example, "*net*".

-top_net_of_hierarchical_group

Keeps only the top net of a hierarchical group. When more than one hierarchical net of the same group is specified (local nets at various hierarchical levels of the same physical net), only the net closest to the top of the hierarchy is saved with the collection. In the case of multiple nets at the same level, the first net specified is kept. Although this option can be used when there are multiple

nets specified, it is best used in combination with **-segments** and with a single net.

-segments

Returns all global segments for given nets. This modifies the initial search which matches *patterns* or *objects*. It is applied before the **-top_net_of_hierarchical_group** is determined. For each net, all global segments which are part of that net will be added to the result collection. Global net segments are all those physically connected across all hierarchical boundaries. Although this option can be used with other options and when there are multiple nets specified, it is best used with a single net. When combined with the **-top_net_of_hierarchical_group** option, you can isolate the highest net segment of a physical net.

-boundary_type lower | **upper** | **both**

Specifies what to do when getting nets of boundary pins. This option requires the **-of_objects** option. Allowed values are lower, upper, and both, meaning the net inside the hierarchical block, outside the hierarchical block, or both nets, respectively. The option has no meaning for non-hierarchical pins. Note that the "upper" value is less useful, as getting pins of a hierarchical net will return the pin outside the hierarchical block, and a subsequent **get_nets** commands find the upper hierarchical net. Using "upper" on a hierarchical pin is the same as omitting the option.

-hsc ***separator***

Specifies a separator character. The default is /. Valid values are /, @, ^, #, ., and |.

-of_objects ***objects***

Creates a collection of nets connected to the specified objects. Each object is either a named pin, a pin collection, a named cell, cell collection, a named net or a net collection, a named via or a via collection, a shape collection, a routing corridor, a bundle, a net bus,a named via_matrix or a via_matrix collection.

The **-of_objects** and *patterns* arguments are mutually exclusive; you must specify one, but not both. In addition, **-hierarchical** and **-of_objects** can be used together only when **-physical_context** is also used and the *objects* argument to **-of_objects** are nets.

-include_shielded

Specifies to include the shielded nets when **-of_objects** contains shapes. It is only valid when **-of_objects** contains shapes.

This option is mutually exclusive with **-shielded_only**.

-shielded_only

Specifies to get the shielded nets only when **-of_objects** contains shapes. It is only valid when **-of_objects** contains shapes.

This option is mutually exclusive with **-include_shielded**.

patterns

Matches net names against patterns. Patterns can include the wildcard characters "*" and "?", or regular expressions based on the **-regexp** option. Patterns can also include collections of type net. Patterns can also include subscript patterns, but not in conjunction with wildcard and regular expressions pattern matching. *patterns* and **-of_objects** are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_nets** command creates a collection of nets in the current design, relative to the current instance that match certain criteria. The command returns a collection if any nets match the *patterns* or *objects* and pass the filter (if specified). If no objects matched the criteria, the empty collection is returned.

If any *patterns* (or *objects*) fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

Subscript patterns are also supported, but cannot be used in conjunction with wildcard or regular expression pattern matching. Bus subscripting is specified with the format "name[begin_index:end_index:step_n]", and returns every nth bus net inclusively between the begin_index and end_index indexes. Note that the "[]" characters must be escaped, because they are Tcl special characters (e.g., A\[0:15:2\] or {A[0:15:2]}). Name expansion subscripting is specified with the format "name(begin_index:end_index:step_n)", and returns every nth net inclusively between the begin and end indexes, which has a matching name without the delimiters characters (that is, net names do not actually contain the '(' and ')' characters). Note that ":step_n" is optional. The default step is 1. Multiple "begin_index:end_index:step_n" patterns can be specified, separated with a ',' character.

You can use the **get_nets** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_nets** result to a variable.

When issued from the command prompt, **get_nets** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_nets** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_nets** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the nets that begin with 'NET' in block 'block1'. Although the output looks like a list, it is just a display.

```
prompt> get_nets block1/NET
{"block1/NET1QNX", "block1/NET2QNX"}
```

The following example queries nets by using bus subscripting.

```
prompt> get_nets {A[0:1,3:7:2]}
{"A[0]", "A[1]", "A[3]", "A[5]", "A[7]"}
```

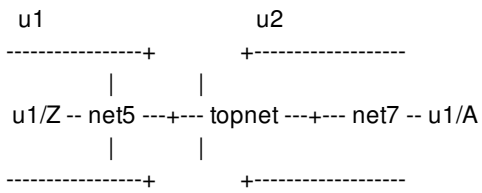
The following example queries nets by using name expansion subscripting.

```
prompt> get_nets B(0:4:2)C(1:2,5)
{"B0C1", "B0C2", "B0C5", "B2C1", "B2C2", "B2C5",
 "B4C1", "B4C2", "B4C5"}
```

The following example queries the nets connected to a collection of pins returned by the **get_pins** command.

```
prompt> current_instance block1
block1
prompt> set pinsel [get_pins {o_reg1/QN o_reg2/QN}]
{"o_reg1/QN", "o_reg2/QN"}
prompt> query_objects [get_nets -of_objects $pinsel]
{"NET1QNX", "NET2QNX"}
```

This example shows how to use the **-top_net_of_hierarchical_group** and **-boundary_type** options. Given the following circuit:



There are hierarchical cells u1 and u2 at this level, connected by a net *topnet*. Within u1 is a pin u1/Z driving *net5*, and within u2 is a pin u1/A being driven by *net7*. These three nets are physically the same net. Assume these are the only nets in the design. Notice the difference in the results of the following two **get_nets** commands:

```
prompt> get_nets * -hierarchical
{topnet u1/net5 u2/net7}
prompt> get_nets * -hierarchical -top_net_of_hierarchical_group
{topnet}
prompt> get_nets -physical_context
{topnet}
prompt> get_nets -physical_context -of_objects [get_nets u2/net7]
{topnet}
```

Assume that at the top level, topnet is connected to pins u2/in and u1/out. Here are some examples of **-boundary_type**. Notice now in this case, the "upper" type does not add value.

```
prompt> get_nets -of_objects u2/in
{topnet}
prompt> get_nets -of_objects u2/in -boundary_type upper
{topnet}
prompt> get_nets -of_objects u2/in -boundary_type lower
{u2/net7}
prompt> get_nets -of_objects u2/in -boundary_type both
{u2/net7 topnet}
prompt> get_nets -boundary_type lower -of_objects \
  [get_pins -of_objects [get_nets topnet]]
{u1/net5 u2/net7}
```

SEE ALSO

collections(2)
filter_collection(2)
get_pins(2)
link_block(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_net_buses

Creates a collection of net buses from the current design. You can assign these net buses to a variable or pass them into another command.

SYNTAX

collection **get_net_buses**
[-design *design*]
[-hierarchical]
[-physical_context]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
patterns | -of_objects *objects*

Data Types

design string or collection
expression string
exact_count integer
minimum_count integer
patterns list
objects string or collection

ARGUMENTS

-design *design_name*

Specifies the top-level design for finding objects. If this is not specified, objects are found in the current design.

-hierarchical

Searches for bus nets within the hierarchy relative to the current instance. The full name of the object at a particular level of hierarchy must match *patterns*. The search is similar to the UNIX **find** command. For example, if there is a bus net block1/bus1, a hierarchical search would find it using "bus1"

-physical_context

This option is needed when a cell object is provided as input to the **-of_objects** option. The net buses of the cells are returned, if the cell is a physical hierarchical cell.

-filter *expression*

Filters the collection with *expression*. For any net buses that match *patterns*, the expression is evaluated based on the net's attributes. If the expression evaluates to true, the cell is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Specifies that the *patterns* argument is a real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. Use this option when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-of_objects *objects*

Creates a collection of net buses connected to the specified objects. Each object is a named cell, cell collection, named net, net collection, named block or block collection. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

patterns

Matches net names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type net. Patterns can also include subscript patterns, but not in conjunction with wildcard and regular expressions pattern matching. *patterns* and **-of_objects** are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_net_buses** command creates a collection of net buses in the current design that match the specified criteria. The command returns a collection if any net buses match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the **regexp** Tcl command. When using **-regexp**, take care in the way you quote the

patterns and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_net_buses** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_net_buses** result to a variable.

When issued from the command prompt, **get_net_buses** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_net_buses** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_net_buses** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page. In addition, refer to the man pages for **all_inputs** and **all_outputs**, which also create collections of net buses.

EXAMPLES

The following example queries all input net buses beginning with 'bus'. Although the output looks like a list, it is just a display.

```
prompt> get_net_buses "bus*"
{"bus1", "bus2"}
```

SEE ALSO

all_inputs(2)
all_outputs(2)
collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_net_estimation_rules

Returns a net_estimation_rule object as defined by the **set_net_estimation_rule** command.

SYNTAX

int **get_net_estimation_rules**
[pattern] [-filter *expression*]
[-quiet]
[-cell *cell*]

Data Types

expression string
pattern string
cell collection

ARGUMENTS

patterns

Matches rule names against patterns. Patterns can include the wildcard characters "*" and "?". If no pattern is specified, it defaults to "*". Patterns can also include collections of type net_estimation_rule.

-filter *expression*

Filters the collection with *expression*. For any rules that match *patterns* (or *objects*), the expression is evaluated based on the attributes of the rule. If the expression evaluates to true, the rule is included in the result.

-quiet

Suppresses warning and error messages if no matching net_estimation_rules are found.

-cell *cell*

Specifies the physical cell from which to find the net estimation rule.

By default, the current block is searched for the specified net estimation rule(s)..

DESCRIPTION

This command returns net_estimation_rule objects as previously defined by the **set_net_estimation_rule** command. The object is useful for querying many attributes, which can be specified or derived automatically by the tool. For example, you can specify a

buffer chain of size 12 buffers and query the calculated attributes of the buffer chain; the command returns the delay of the buffer chain. Use this command in Tcl scripts to explore the characteristics of your technology.

Alternatively you can use **report_net_estimation_rules** to examine the attribute values.

NET ESTIMATION RULES AND HIERARCHY

Net estimation rules can be created in any physical block at any level of the hierarchy and multiple blocks can have different rules with the same name. However, rules are often stored/retrieved by name, and doing this can cause confusion as to which rule is being used by a given command. Generally, the "current" (top) block's list of rules is searched by commands such as *estimate_timing*. This means that if you define rule R1 in "top" and rule R1 in block "B1", if you make block B1 the current block, commands will use B1's R1. When you go back to top, top's R1 is used. To avoid confusion, use globally unique rule names and define them at the top - they will be propagated to child blocks during distributed commands as necessary. Rules in child blocks that have the same name as rules in top may also be overwritten during distributed commands.

EXAMPLES

The following example retrieves all attributes for the "default" rule.

```
prompt> report_attributes -application [get_net_estimation_rules default]
```

The following example returns the horizontal buffer spacing for the "M5_M6" rule.

```
prompt> get_attribute [get_net_estimation_rules M5_M6] h_buffer_spacing
```

SEE ALSO

report_net_estimation_rules(2)
set_net_estimation_rule(2)

get_nets

Creates a collection of nets from the netlist. You can assign these nets to a variable or pass them to another command.

SYNTAX

collection **get_nets**
[-design *design*]
[-hierarchical]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-physical_context]
[-top_net_of_hierarchical_group]
[-segments]
[-boundary_type lower | upper | both]
[-hsc *separator*]
[-include_shielded]
[-shielded_only]]
patterns | -of_objects *objects*

Data Types

design string
expression string
exact_count integer
minimum_count integer
separator character
patterns list
objects list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this option is not specified, objects will be found in the current design.

-hierarchical

Searches for nets within the hierarchy relative to the current instance, when used without the **-physical_context** option. The full

name of the object at a particular level of hierarchy must match *patterns*. The search is similar to the UNIX **find** command. For example, if there is a net u1/muxsel, a hierarchical search would find it using "muxsel". **-hierarchical** and **-of_objects**, can be used together only when **-physical_context** is also used and the *objects* argument to **-of_objects** are nets. When used together with **-physical_context** option, the search is performed level-by-level, where each level is a physical hierarchy. So, if u1 is a logical hierarchical instance, then the search will **not** find the net "muxsel". However, search for "*muxsel", would find the net physical net u1/muxsel.

-filter *expression*

Filters the collection with *expression*. For any nets that match *patterns* (or *objects*), the expression is evaluated based on the attributes of the cell. If the expression evaluates to true, the net is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Considers the *patterns* argument as true regular expressions rather than a simple wildcard pattern. This option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-physical_context

Searches only for physical nets. This option can also be used with **-of_objects**. If the *objects* argument to **-of_objects** is a pin, the physical net connected to the pin is returned. If the *objects* argument to **-of_objects** is a cell, a collection of physical nets connected to the pins of the cell is returned. If *objects* argument to **-of_objects** is a net, the physical net corresponding to the net is returned. If **-of_objects** is not used, the physical nets matching the specified *patterns* are returned. When **-hierarchical** is used, the patterns are matched against the name of the physical nets, which typically correspond to full hierarchical name of logical nets. Therefore, a search for "net*" will only find nets in the topmost logical hierarchy, matching the pattern "net*". To find physical nets in lower levels of the logical hierarchy, prefix the expression with an asterisk (*), for example, "*net*".

-top_net_of_hierarchical_group

Keeps only the top net of a hierarchical group. When more than one hierarchical net of the same group is specified (local nets at various hierarchical levels of the same physical net), only the net closest to the top of the hierarchy is saved with the collection. In the case of multiple nets at the same level, the first net specified is kept. Although this option can be used when there are multiple

nets specified, it is best used in combination with **-segments** and with a single net.

-segments

Returns all global segments for given nets. This modifies the initial search which matches *patterns* or *objects*. It is applied before the **-top_net_of_hierarchical_group** is determined. For each net, all global segments which are part of that net will be added to the result collection. Global net segments are all those physically connected across all hierarchical boundaries. Although this option can be used with other options and when there are multiple nets specified, it is best used with a single net. When combined with the **-top_net_of_hierarchical_group** option, you can isolate the highest net segment of a physical net.

-boundary_type lower | **upper** | **both**

Specifies what to do when getting nets of boundary pins. This option requires the **-of_objects** option. Allowed values are lower, upper, and both, meaning the net inside the hierarchical block, outside the hierarchical block, or both nets, respectively. The option has no meaning for non-hierarchical pins. Note that the "upper" value is less useful, as getting pins of a hierarchical net will return the pin outside the hierarchical block, and a subsequent **get_nets** commands find the upper hierarchical net. Using "upper" on a hierarchical pin is the same as omitting the option.

-hsc ***separator***

Specifies a separator character. The default is /. Valid values are /, @, ^, #, ., and |.

-of_objects ***objects***

Creates a collection of nets connected to the specified objects. Each object is either a named pin, a pin collection, a named cell, cell collection, a named net or a net collection, a named via or a via collection, a shape collection, a routing corridor, a bundle, a net bus,a named via_matrix or a via_matrix collection.

The **-of_objects** and *patterns* arguments are mutually exclusive; you must specify one, but not both. In addition, **-hierarchical** and **-of_objects** can be used together only when **-physical_context** is also used and the *objects* argument to **-of_objects** are nets.

-include_shielded

Specifies to include the shielded nets when **-of_objects** contains shapes. It is only valid when **-of_objects** contains shapes.

This option is mutually exclusive with **-shielded_only**.

-shielded_only

Specifies to get the shielded nets only when **-of_objects** contains shapes. It is only valid when **-of_objects** contains shapes.

This option is mutually exclusive with **-include_shielded**.

patterns

Matches net names against patterns. Patterns can include the wildcard characters "*" and "?", or regular expressions based on the **-regexp** option. Patterns can also include collections of type net. Patterns can also include subscript patterns, but not in conjunction with wildcard and regular expressions pattern matching. *patterns* and **-of_objects** are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_nets** command creates a collection of nets in the current design, relative to the current instance that match certain criteria. The command returns a collection if any nets match the *patterns* or *objects* and pass the filter (if specified). If no objects matched the criteria, the empty collection is returned.

If any *patterns* (or *objects*) fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

Subscript patterns are also supported, but cannot be used in conjunction with wildcard or regular expression pattern matching. Bus subscripting is specified with the format "name[begin_index:end_index:step_n]", and returns every nth bus net inclusively between the begin_index and end_index indexes. Note that the "[]" characters must be escaped, because they are Tcl special characters (e.g., A\[0:15:2\] or {A[0:15:2]}). Name expansion subscripting is specified with the format "name(begin_index:end_index:step_n)", and returns every nth net inclusively between the begin and end indexes, which has a matching name without the delimiters characters (that is, net names do not actually contain the '(' and ')' characters). Note that ":step_n" is optional. The default step is 1. Multiple "begin_index:end_index:step_n" patterns can be specified, separated with a ',' character.

You can use the **get_nets** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_nets** result to a variable.

When issued from the command prompt, **get_nets** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_nets** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_nets** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the nets that begin with 'NET' in block 'block1'. Although the output looks like a list, it is just a display.

```
prompt> get_nets block1/NET
{"block1/NET1QNX", "block1/NET2QNX"}
```

The following example queries nets by using bus subscripting.

```
prompt> get_nets {A[0:1,3:7:2]}
{"A[0]", "A[1]", "A[3]", "A[5]", "A[7]"}
```

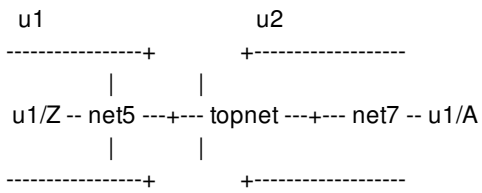
The following example queries nets by using name expansion subscripting.

```
prompt> get_nets B(0:4:2)C(1:2,5)
{"B0C1", "B0C2", "B0C5", "B2C1", "B2C2", "B2C5",
 "B4C1", "B4C2", "B4C5"}
```

The following example queries the nets connected to a collection of pins returned by the **get_pins** command.

```
prompt> current_instance block1
block1
prompt> set pinsel [get_pins {o_reg1/QN o_reg2/QN}]
{"o_reg1/QN", "o_reg2/QN"}
prompt> query_objects [get_nets -of_objects $pinsel]
{"NET1QNX", "NET2QNX"}
```

This example shows how to use the **-top_net_of_hierarchical_group** and **-boundary_type** options. Given the following circuit:



There are hierarchical cells u1 and u2 at this level, connected by a net *topnet*. Within u1 is a pin u1/Z driving *net5*, and within u2 is a pin u1/A being driven by *net7*. These three nets are physically the same net. Assume these are the only nets in the design. Notice the difference in the results of the following two **get_nets** commands:

```
prompt> get_nets * -hierarchical
{topnet u1/net5 u2/net7}
prompt> get_nets * -hierarchical -top_net_of_hierarchical_group
{topnet}
prompt> get_nets -physical_context
{topnet}
prompt> get_nets -physical_context -of_objects [get_nets u2/net7]
{topnet}
```

Assume that at the top level, topnet is connected to pins u2/in and u1/out. Here are some examples of **-boundary_type**. Notice now in this case, the "upper" type does not add value.

```
prompt> get_nets -of_objects u2/in
{topnet}
prompt> get_nets -of_objects u2/in -boundary_type upper
{topnet}
prompt> get_nets -of_objects u2/in -boundary_type lower
{u2/net7}
prompt> get_nets -of_objects u2/in -boundary_type both
{u2/net7 topnet}
prompt> get_nets -boundary_type lower -of_objects \
  [get_pins -of_objects [get_nets topnet]]
{u1/net5 u2/net7}
```

SEE ALSO

collections(2)
filter_collection(2)
get_pins(2)
link_block(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_num_scan_chains

Returns the number of chains in a given test mode.

SYNTAX

list **get_num_scan_chains**
[-test_mode *mode_name*]

Data Types

mode_name string

ARGUMENTS

-test_mode *mode_name*

Returns the number of chains for the specified test mode. By default, the tool reports on the current test mode. If this option is not specified and there is no current test mode, the command returns an error.

DESCRIPTION

This command returns the number of scan chains for the specified test mode, or the current test mode if a mode is not specified.

EXAMPLES

The following example reports the number of scan chains for some test modes.

```
prompt> get_num_scan_chains -test_mode Internal_scan
8
prompt> get_num_scan_chains -test_mode ScanCompression_mode
32
```

SEE ALSO

current_test_mode(2)
get_scan_cell_names(2)
report_dft(2)
set_dft_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(2)

get_object_by_id

Creates a collection of objects from the current design relative to the current instance with the input of object_id(s). You can assign these objects to a variable or pass them into another command.

SYNTAX

collection **get_object_by_id**
object_list

Data Types

object_list list

ARGUMENTS

object_list

Need to pass a list of of object_ids to the command, otherwise the command reports nothing. The object types supported are cell, pin, port, lib_cell and net.

DESCRIPTION

The **get_object_by_id** command creates a collection of objects in the current design, relative to the current instance, that match certain criteria. The object types supported for the object_id attribute list passed to the command are cell/port/pin/lib_cell/net. If the object_id doestn't point to an object, then the object_id will be skipped. If no objects are found with the list of object_ids passed, the command doesn't report anything.

EXAMPLES

The following example tries to get the object collection given object_ids list passed to the command.

```
prompt>get_attribute [get_cells {U_0 r1_reg}] object_id
260628496 260628692
prompt> get_object_by_id {260628496 260628692}
{U_0 r1_reg}
```

SEE ALSO

collections(2)
filter_collection(2)
get_attribute(2)

get_object_name

Gets the name of the objects in a collection.

SYNTAX

string **get_object_name** *collection*

string *collection*

ARGUMENTS

collection

Specifies the collection.

DESCRIPTION

The **get_object_name** command is a convenient way to get the *full_name* attribute from any object or collection of objects.

EXAMPLES

The following example shows how to use **get_object_name**.

```
prompt> get_object_name [get_cells i1]
i1
prompt> get_attribute [get_cells i1] full_name
i1
prompt> get_object_name [get_cells i*]
i1 i2 i3
```

SEE ALSO

collections(2)
get_attribute(2)

get_object_occurrences

Creates a collection of all occurrences of a physical object in the hierarchy of the top design.

SYNTAX

collection **get_object_occurrences**
[-quiet]
[*object*]

Data Types

object list

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

object

Specifies name or collection of a single physical object in the top design's context.

DESCRIPTION

This command creates a homogeneous collection of all occurrences of the physical object given a handle to one of the physical object. If there is only one occurrence of the physical object than it returns the same. You can use the **get_object_occurrences** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable. See the **collections** command man page for information about working with collections.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example get all occurrences of the cell in the hierarchy of the top design given a single occurrence of the cell as input.

```
prompt> get_object_occurrences [ get_cells I_ORCA_TOP/I_BLENDER_2/U1664]
{I_ORCA_TOP/I_BLENDER_2/U1664 I_ORCA_TOP/I_BLENDER_5/U1664}
```

SEE ALSO

collections(2)
filter_collection(2)
query_objects(2)
shell.common.collection_result_display_limit(3)

get_objects_by_location

Creates a homogeneous or heterogeneous collection of physical objects by using a region-based search criteria.

SYNTAX

collection **get_objects_by_location**
[-design *design*]
[-filter *expression*]
[-quiet]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-classes *classes*]
[-hierarchical]
[-include_fill_shapes]
[-at *point*]
[-within { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects]
[-touching { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects]
[-intersect { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects]

Data Types

design string
expression string
exact_count int
minimum_count int
classes list
point coordinates
llx float
lly float
urx float
ury float
x float
y float
geometric_objects collection

ARGUMENTS

-design *design*

Specifies the top-level design in which to find objects. If this is not specified, the command searches for objects in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on attributes of the physical object. You can determine the attributes by using the **list_attributes** command. If the expression evaluates to **true**, the object is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-expect *exact_count*

Specifies an expected number of objects to return. If the command finds a different number of objects, the command raises a Tcl error and terminates.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, the command raises a Tcl error and terminates.

-classes *classes*

Specifies the physical objects for which the search is performed. By default, all the supported physical objects are searched. The supported values are:

- "*routing_blockage*"
- "*placement_blockage*"
- "*bound*"
- "*edit_group*"
- "*fill_cell*"
- "*io_guide*"
- "*io_ring*"
- "*pg_region*"
- "*pin_blockage*"
- "*pin_guide*"
- "*routing_guide*"
- "*shape*"
- "*shape_pattern*"
- "*via*"
- "*via_region*"
- "*via_matrix*"
- "*site_row*"

- "*track*"
- "*voltage_area*"
- "*routing_corridor*"
- "*rp_group*"
- "*rp_blockage*"
- "*site_array*"
- "*cell*"
- "*port*"
- "*net*"
- "*terminal*"
- "*lib_pin*"
- "*pin*"
- "*bound_shape*"
- "*voltage_area_shape*"
- "*routing_corridor_shape*"

-hierarchical

Searches for physical objects in the full hierarchy below the top-level design. By default, the search is limited to the top-level design.

-include_fill_shapes

Search for fill shapes in hierarchy below the top-level design. By default, fill shapes search is limited to top-level design.

-at *point*

Searches all physical objects of given classes whose region completely contains the input point. This option is mutually exclusive with the **-within**, **-touching** and **-intersect** options. At least one of **-at**, **-within**, **-touching** or **-intersect** must be specified.

-within { {{*llx lly*} {*urx ury*}} | {{*x y*} {*x y*} {*x y*} {*x y*}...} } | geometric_objects

Searches all physical objects of the given classes whose region is completely contained by the specified regions. All objects which are abut/touching from inside and completely contained also get selected.

When the **design.enable_icc_region_query** application option is set to *true*, only the objects which are completely contained but not abut/touching from inside get selected.

The region can be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates (i.e., {*llx lly*} {*urx ury*}). A polygon is specified by its points (i.e., {*x y*} {*x y*} {*x y*} {*x y*}...).

Polygons can also be specified as the combined area of a heterogeneous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

Region specified can be a disjointed rectangle or polygon.

This option is mutually exclusive with the **-at** and **-touching** options. At least one of **-at**, **-within** , **-touching** or **-intersect** must be

specified.

-touching { {{*llx lly*} {*urx ury*}} | {{*x y*} {*x y*} {*x y*} {*x y*}...} } | geometric_objects

Searches all physical objects of the given classes whose region is completely contained by the specified region. All objects which are touching from inside and completely contained also get selected.

The region may be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates (i.e., {*llx lly*} {*urx ury*}). A polygon is specified by its points (i.e., {*x y*} {*x y*} {*x y*} {*x y*}...).

Polygons may also be specified as the combined area of a heterogeneous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

Region specified can be a disjointed rectangle or polygon.

This option is mutually exclusive with the **-at** , **-within** and **-intersect** options. At least one of **-at**, **-within** , **-touching** or **-intersect** must be specified.

-intersect { {{*llx lly*} {*urx ury*}} | {{*x y*} {*x y*} {*x y*} {*x y*}...} } | geometric_objects

Searches all physical objects of the given classes whose region intersects the specified region. The certain portion of the region should be completely inside and certain portion outside of the input window to pass this test.

When the **design.enable_icc_region_query** application option is set to *true*, even the objects which are abut/touching from either inside or outside also get selected.

Region specified can be a disjointed rectangle or polygon.

This option is mutually exclusive with the **-at** and **-touching** options. At least one of **-at**, **-within** , **-touching** or **-intersect** must be specified.

DESCRIPTION

This command performs a physical search on specific classes of physical objects and returns a homogeneous/heterogeneous collection of the queried physical objects. A particular invocation of the command queries for the input user object classes. All the objects of the given classes which satisfy the search parameters and other filtering parameters are returned as a homogeneous/heterogeneous collection. The collection of homogeneous/heterogeneous objects is returned with respect to the context of the top-level design or to the design explicitly specified with **-design**. The input point or region is specified with respect to the top-level design coordinate space or the coordinate space of the design specified with **-design**. By default, the command searches for all supported physical objects for a point or region, unless the physical object classes are explicitly provided. In most cases, the command returns a heterogeneous collection. If no object matches the search criteria, it returns an empty string. You can use the **get_objects_by_location** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable. See the **collections** command man page for information about working with collections.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns the cells in the top-level design that completely contain the point {830 70}.

```
prompt> get_objects_by_location -classes {cell} -at {830 70}
{MEM1}
```

SEE ALSO

collections(2)
filter_collection(2)
query_objects(2)
design.enable_icc_region_query(3)
shell.common.collection_result_display_limit(3)

get_output_delays

Creates a collection of output delays in the current scenario. You can assign these output delays to a variable or pass them to another command.

SYNTAX

collection **get_output_delays**
-of_objects *objects*
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-filter *expression*]
[-quiet]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]

Data Types

mode_list collection
corner_list collection
scenario_list collection
objects collection
expression string
exact_count int
minimum_count int

ARGUMENTS

-of_objects *objects*

Creates a collection of output delays defined on the specified objects. Each object is either a named pin, a pin collection, a named port or port collection.

-modes *mode_list*

Specifies the scenarios from which to retrieve the objects. If this option is given, all scenarios of the specified modes will be used. The **-scenarios** option must not be specified together with the **-modes** option. If both **-modes** and **-corners** are specified, the command uses all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners**.

-corners *corner_list*

Specifies the scenarios from which to retrieve the objects. If this option is given, all scenarios of the specified corners and the current mode will be used. The **-scenarios** option must not be specified together with the **-corners** option. If both **-modes** and **-corners** are specified, the command uses all scenarios whose mode is specified by **-modes** and whose corner is specified by **-**

corners.

-scenarios ***scenario_list***

Specifies the scenarios from which to retrieve the objects. The **-modes** or **-corners** option must not be specified together with the **-scenarios** option.

-filter ***expression***

Filters the collection with *expression*. For any objects that match, the expression is evaluated based on the object's attributes. If the expression evaluates to true, the object is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-expect ***exact_count***

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least ***minimum_count***

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

DESCRIPTION

The **get_output_delays** command creates a collection of output delays matching *patterns* in the current scenario and which pass the filter, if specified. If no objects match the criteria, the empty string is returned.

Output delay data is managed on a per-scenario basis. For designs with multiple scenarios, you can specify scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command applies to the current scenario. The tool issues an error if there is no current scenario. If the **-scenarios** option is given, the command applies to all of the specified scenarios.

If the **-modes** option is given, the command applies to all scenarios of the specified modes. The tool issues an error if none of the specified modes have any scenarios.

If the **-corners** option is given, the command applies to all scenarios of the specified corners and the current mode. The tool issues an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are given, the command applies to all scenarios of the specified corners and the specified modes. The tool issues an error if there are no such scenarios. It is an error to specify **-scenarios** with **-modes** or **-corners**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example sets the inDel variable to the collection of output delays defined on ports IN* in the current scenario.

```
prompt> set inDel [get_output_delays -of_objects [get_ports IN*]]
```

SEE ALSO

collections(2)
set_output_delay(2)

get_overlap_blockages

Creates a collection of overlap blockages from the current design.

SYNTAX

collection **get_overlap_blockages**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns* | -of_objects *of_objects*]

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns string
of_objects list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects are found in the current design.

-filter *expression*

Filters the collection with *expression*. For any overlap blockages that match *patterns*, the expression is evaluated based on the overlap blockage's attributes. If the expression evaluates to true, the blockage is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the

=~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for overlap blockages level-by-level relative to the current instance. This option is useful only with *patterns* and not with **-of_objects**.

-of_objects *of_objects*

Creates a collection containing the overlap blockages of the specified cells or blocks. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches the overlap blockage names in the current design against the specified patterns. You can specify the patterns by using the following formats:

- ***** (asterisk) or **OB_*** indicates all overlap blockages
- **OB_77** indicates one overlap blockage whose object_id is 77

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses asterisk (*) as the pattern.

DESCRIPTION

This command returns a collection of overlap blockages of the current design that meet the selection criteria. You can use this command to query the overlap blockages inside the database by area, by point, or by region. You can also use this command to query overlap blockages by filtering their attributes.

EXAMPLES

The following example returns the overlap blockages whose area is larger than 1000.

```
prompt> get_overlap_blockages * -filter "area > 1000"
{"OB_4683"}
```

The following example returns all overlap blockages that begin with "OB_".

```
prompt> get_overlap_blockages "OB_*"
{"OB_4683 OB_5389"}
```

The following three commands show how to use **get_attribute** to query the value of the overlap blockage attributes.

```
prompt> get_attribute [get_overlap_blockages OB_4683] overlap_layer_name
overlap

prompt> get_attribute [get_overlap_blockages OB_4683] bbox
{0.000 0.000{ {100.000 100.000}

prompt> get_attribute [get_overlap_blockages OB_4683] area
10000.000000
```

To get a complete report of all the attribute values of an overlap blockage, use the **report_attributes** command, as shown in the following example.

```
prompt> report_attributes -application [get_overlap_blockages OB_0]
Design   Object   Type       Attribute Name     Value
------------------------------------------------------------------------
CORE     OB_0     area       area               3.6864
CORE     OB_0     rect       bbox               {0.0000 0.0000}
  {2.5600 1.4400}
CORE     OB_0     coord_list boundary           {0.0000 0.0000}
  {0.0000 1.4400} {2.5600 1.4400} {2.5600 0.0000}
CORE     OB_0     string     full_name          OB_0
CORE     OB_0     string     name               OB_0
CORE     OB_0     string     object_class       overlap_blockage
CORE     OB_0     string     overlap_layer_name overlap
CORE     OB_0     collection parent_block       std_lib:CORE.design
```

SEE ALSO

collections(2)
get_attribute(2)
report_attributes(2)

get_parasitic_techs

Get the parasitic tech object from a library.

SYNTAX

get_parasitic_techs
[-library *library_object*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *objects*]

Data Types

string *expression*
list *objects*

ARGUMENTS

-library

to specify one library object from other get command. If no library is specified, all open libraries will be searched to get parasitic techs.

-filter *expression*

Filters the collection with *expression*. For any parasitic_tech that match *patterns* (or *objects*) the expression is evaluated based on the parasitic_tech's attributes. If the expression evaluates to true, the parasitic tech is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modify the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *objects*

Creates a collection of libraries that contain the specified objects. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

patterns

Matches library names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type lib.

DESCRIPTION

The **get_parasitic_techs** command creates a collection of parasitic_techs in the specified library, that match certain criteria. The command returns a collection if any parasitic_techs match the *patterns* or *objects* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

If any *patterns* (or *objects*) fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_parasitic_techs** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_parasitic_techs** result to a variable.

When issued from the command prompt, **get_parasitic_techs** will report the itf_technology_name if found a match.

The "implicit query" property of **get_parasitic_techs** provides a fast, simple way to display parasitic_techs in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_parasitic_techs** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example will get a parasitic tech from library r4000.

prompt> **get_parasitic_techs -filter "itf_technology_name == crn40lp_1p06m+alrdl_4x1u_cworst"**

SEE ALSO

filter_collection(2)
read_parasitic_tech(2)

get_path_group

Creates a collection of path groups from the current design. You can assign these path groups to a variable or pass them into another command.

SYNTAX

collection **get_path_groups**
[-design *design*]
[-mode *mode*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns*]

Data Types

design	collection
mode	collection
expression	string
exact_count	int
minimum_count	int
patterns	list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-mode *mode*

Specifies the mode for finding objects. If this is not specified, objects will be found in the current mode.

-filter *expression*

Filters the collection with *expression*. For any objects that match, the expression is evaluated based on the object's attributes. If the expression evaluates to true, the object is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches path group names against patterns. Patterns can include the wildcard characters "*" and "?".

DESCRIPTION

The **get_path_groups** command creates a collection of path groups from the current design that match certain criteria. The command returns a collection if any path groups match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Path groups control the optimization cost function and also affect timing reports.

You can use the **get_path_groups** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_path_groups** result to a variable.

When issued from the command prompt, the **get_path_groups** command behaves as though the **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of the **get_path_groups** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_path_groups** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-mode** option.

EXAMPLES

The following generates a timing report for each path group matching "Z*".

```
prompt> foreach_in_collection grp [get_path_groups Z*] { \
  report_timing -group $grp \
}
```

SEE ALSO

collections(2)
foreach_in_collection(2)
group_path(2)
query_objects(2)
report_path_groups(2)
shell.common.collection_result_display_limit(3)

get_path_groups

Creates a collection of path groups from the current design. You can assign these path groups to a variable or pass them into another command.

SYNTAX

collection **get_path_groups**
[-design *design*]
[-mode *mode*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns*]

Data Types

design	collection
mode	collection
expression	string
exact_count	int
minimum_count	int
patterns	list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-mode *mode*

Specifies the mode for finding objects. If this is not specified, objects will be found in the current mode.

-filter *expression*

Filters the collection with *expression*. For any objects that match, the expression is evaluated based on the object's attributes. If the expression evaluates to true, the object is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches path group names against patterns. Patterns can include the wildcard characters "*" and "?".

DESCRIPTION

The **get_path_groups** command creates a collection of path groups from the current design that match certain criteria. The command returns a collection if any path groups match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Path groups control the optimization cost function and also affect timing reports.

You can use the **get_path_groups** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_path_groups** result to a variable.

When issued from the command prompt, the **get_path_groups** command behaves as though the **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of the **get_path_groups** command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_path_groups** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-mode** option.

EXAMPLES

The following generates a timing report for each path group matching "Z*".

```
prompt> foreach_in_collection grp [get_path_groups Z*] { \
  report_timing -group $grp \
}
```

SEE ALSO

collections(2)
foreach_in_collection(2)
group_path(2)
query_objects(2)
report_path_groups(2)
shell.common.collection_result_display_limit(3)

get_pg_regions

Creates a collection of PG regions from the current design.

SYNTAX

collection **get_pg_regions**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design string *expression* string *exact_count* integer *minimum_count* integer *objects* collection *patterns* string *region* list *point* list

ARGUMENTS

-design *design*

Specifies the top-level design for finding objects. By default, objects are searched in the current design.

-filter *expression*

Filters the collection with *expression*. For any PG regions that match *patterns*, the expression is evaluated based on the attributes for the PG region. If the expression evaluates to true, the clock is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. This option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Enforces exact pattern matching and disables simple pattern matching. Use this option when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer than the specified number of objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Requires that each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-of_objects *of_objects*

Specifies the edit groups to use to search for PG regions.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

patterns

Matches PG region names against patterns. Patterns can include the asterisk wildcard character (*) and the question mark character (?). For more information about using and escaping wildcards, see the **wildcards** man page.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all pg_regions at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-within *region*

Creates a collection that contains all pg_regions that are completely inside the specified region and doesn't include pg_regions which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-touching *region*

Creates a collection that contains all pg_regions that are completely inside the specified region and the pg_regions which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-intersect *region*

Creates a collection that contains all pg_regions which are abut/touching from inside or outside to the specified region and the pg_regions whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

DESCRIPTION

This command returns a collection of PG regions in the current design that meet the selection criteria. You can use this command to query PG regions by filtering their attributes.

EXAMPLES

The following example returns all PG regions beginning with PGR_.

```
prompt> get_pg_regions "PGR_*"
{"PGR_4683 PGR_5389"}
```

The following example uses the **get_attribute** command to query the value of the PG region attributes.

```
prompt> get_attribute [get_pg_regions PGR_1123] name
PGR_1123

prompt> get_attribute [get_pg_regions PGR_4683] bbox
{0.000 0.000} {100.000 100.000}
```

SEE ALSO

collections(2)
create_pg_region(2)
remove_pg_regions(2)
report_pg_regions(2)

get_pin

Creates a collection of pins from the netlist. You can assign the collection to a variable or pass the collection to another command.

SYNTAX

collection **get_pins**
[-hierarchical]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-leaf]
[-physical_context]
[-design *design_name*]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hsc *separator_character*]
patterns | -of_objects *objects*

Data Types

expression string
design_name string
exact_count integer
minimum_count integer
separator_character string
patterns list
objects list

ARGUMENTS

-hierarchical

Searches for pins level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX **find** command. For example, if there is a pin block1/adder/D[0], a hierarchical search finds it using "adder/D[0]". You cannot use **-hierarchical** with the **-of_objects** option.

-filter *expression*

Filters the collection with *expression*. For any pins that match *patterns* or *objects*, the expression is evaluated based on the attributes of the pin. If the expression evaluates to true, the pin is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as true regular expressions rather than as simple wildcard patterns. This option modifies the behavior of the =~ and !~ filter operators to compare with true regular expressions rather than simple wildcard patterns. The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Performs case-insensitive pattern matching.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The **-exact** and **-regexp** options are mutually exclusive.

-leaf

Includes only pins on leaf cells connected to the specified nets. You can use this option only with **-of_objects**. For any nets in the *objects* argument to **-of_objects**, only pins on leaf cells connected to those nets are included in the collection. In addition, hierarchical boundaries are crossed to find pins on leaf cells.

-physical_context

Searches only for physical pins. This option can also be used with **-of_objects**. For any nets in the *objects* argument to **-of_objects**, only pins on physical cells connected to those nets are included in the collection. In addition, hierarchical boundaries are crossed to find pins on physical cells. For any cells in the *objects* argument to **-of_objects**, pins of the cell are returned, if the cell is physical. If **-of_objects**, is not used than physical pins matching the specified *patterns* are returned.

-design *design_name*

Specifies the name of the design in which to find pins.

-expect *exact_count*

Specifies an exact number of objects to find. If the command finds a different number of objects, the command raises a Tcl error and stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, the command raises a Tcl error and stops.

-expect_each_pattern_matches

Ensures that each pattern in *patterns* matches at least one object. If one or more patterns does not match, the command raises a Tcl error and stops.

-hsc separator_character

Specifies a separator character. The default is /. Valid values are /, @, ^, #, ., and |.

patterns

Matches pin names against patterns. Patterns can include the wildcard characters "*" and "?", or regular expressions, based on the **-regexp** option. Patterns can also include collections of type pin. Patterns can also include subscript patterns, but not in conjunction with wildcard and regular expressions pattern matching. *patterns* and **-of_objects** are mutually exclusive; you must specify one, but not both.

-of_objects *objects*

Creates a collection of pins connected to the specified objects. Each object is a named cell, named net, named stub_chain,

named scan_chain, cell collection, net collection, stub_chain collection or scan_chain collection.

The **-of_objects** and *patterns* options are mutually exclusive; you must specify either option, but not both. In addition, you cannot use the **-hierarchical** option with **-of_objects**.

DESCRIPTION

The **get_pins** command creates a collection of pins in the current design, relative to the current instance, that match certain criteria. The command returns a collection of pins that match the specified *patterns* or *objects*, and pass the filter, if specified. If no objects match the criteria, the empty string is returned.

When used with **-of_objects**, **get_pins** searches for pins connected to any cells or nets specified in *objects*. For net objects, there are two variations of pins that are considered. By default, only pins connected to the net at the same hierarchical level are considered. When combined with the **-leaf** option, only pins connected to the net that are on leaf cells are considered. In this case, hierarchical boundaries are crossed to find pins on leaf cells. Note that **-leaf** has no effect on the pins of cells.

If no *patterns* (or *objects*) match any objects, and the current design is not linked, the design automatically links.

When a cell has bus pins, **get_pins** can find them in several ways. For example, if cell u1 has a bus "A" with indexes 2 to 0, and the verilog.bus_naming_style for your design is "%s[%d]", use "u1/A[*]" as the pattern. You can also find the same three pins with "u1/A" as the pattern.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

Subscript patterns are also supported, but cannot be used in conjunction with wildcard or regular expression pattern matching. Bus subscripting is specified with the format "name[begin_index:end_index:step_n]", and returns every nth bus pin inclusively between the begin and end indices. Note that the "[]" characters must be escaped, because they are TCL special characters (e.g., A\[0:15:2\] or {A[0:15:2]}). Name expansion subscripting is specified with the format "name(begin_index:end_index:step_n)", and returns every nth pin inclusively between the begin and end indices, which has a matching name without the delimiters characters (i.e., pin names do not actually contain the '(' and ')' characters). Note that ":step_n" is optional. The default step is 1. Multiple "begin_index:end_index:step_n" patterns may be specified, separated with a ',' character.

You can use the **get_pins** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_pins** result to a variable.

When issued from the command prompt, **get_pins** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can increase or decrease the number of results that are displayed by setting the **shell.collection_result_display_limit** application option.

The "implicit query" property of **get_pins** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_pins** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the 'CP' pins of cells that begin with 'o'. Although the output looks like a list, it is just a display.

```
prompt> get_pins o*/CP
{"o_reg1/CP", "o_reg2/CP", "o_reg3/CP", "o_reg4/CP"}
```

The following example queries pins using bus subscripting.

```
prompt> get_pins {U1/A[0:1,3:7:2]}
{"U1/A[0]", "U1/A[1]", "U1/A[3]", "U1/A[5]", "U1/A[7]"}
```

The following example queries pins using name expansion subscripting.

```
prompt> get_pins U1/B(0:4:2)C(1:2,5)
{"U1/B0C1", "U1/B0C2", "U1/B0C5", "U1/B2C1", "U1/B2C2", "U1/B2C5",
 "U1/B4C1", "U1/B4C2", "U1/B4C5"}
```

The following example queries the pins connected to the specified cells.

```
prompt> set csel [get_cells o_reg1]
{"o_reg1"}
prompt> query_objects [get_pins -of_objects $csel]
{"o_reg1/D", "o_reg1/CP", "o_reg1/CD", "o_reg1/Q", "o_reg1/QN"}
```

The following example shows the difference between getting local pins of a net and getting the leaf pins of net. In this example, NET1 is connected to i2/a and reg1/QN. Cell i2 is hierarchical. Within i2, port a is connected to the U1/A and U2/A.

```
prompt> get_pins -of_objects [get_nets NET1]
{"i2/a", "reg1/QN"}
prompt> get_pins -leaf -of_objects [get_nets NET1]
{"i2/U1/A", "i2/U2/A", "reg1/QN"}
```

The following example shows how to create a clock using a collection of pins.

```
prompt> create_clock -period 8 -name CLK [get_pins o_reg*/CP]
1
```

SEE ALSO

collection_result_display_limit(3)
collections(2)
create_clock(2)
filter_collection(2)
get_cells(2)
link_design(2)
query_objects(2)
regexp(2)
verilog_options(3)

get_pin_blockages

Creates a collection of pin blockages from the current design.

SYNTAX

collection **get_pin_blockages**
[-filter *expression*]
[-quiet]
[-design *design_name*]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

expression string
design_name string
exact_count int
minimum_count int
patterns list
objects list
region list
point list

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the pin blockage attributes. You can determine the pin blockage attributes by using the **list_attributes** command. If the expression evaluates to **true**, the pin blockage is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-design *design_name*

Specifies the top design for finding objects. If this is not specified, objects are found in the current design.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for pin blockages level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

patterns

Matches pin blockage names against patterns. Patterns can include the wildcard characters the * (asterisk) and ? (question mark) wildcard characters or regular expressions, depending on the use of the **-regexp** option. For more details about using and escaping wildcards, see the **wildcards** man page.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-of_objects *of_objects*

Creates a collection containing the pin blockages associated with the specified objects. The objects can be pins, nets, or ports.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all pin_blockages at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all pin_blockages that are completely inside the specified region and doesn't include pin_blockages which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all pin_blockages that are completely inside the specified region and the pin_blockages which are abut/touching from inside with the specified region. The region boundary can be a rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all pin_blockages which are abut/touching from inside or outside to the specified region and the pin_blockages whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of pin blockages from the current design that match certain criteria. The command returns a collection handle (identifier) if any pin blockages match the value of the *patterns* option and meet the filter criteria (if specified). If no objects match the criteria, the command returns an empty string.

You can use the **get_pin_blockages** command at the command prompt or nest it as an argument to another command, such as **report_pin_blockages**. You can also assign the **get_pin_blockages** result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following returns the pin blockages associated with the specified pins.

```
prompt> get_pin_blockages -of_objects [get_pins cellA/pin*]
{VA1}
```

SEE ALSO

add_to_pin_blockage(2)
create_pin_blockage(2)
remove_from_pin_blockage(2)
remove_pin_blockages(2)
report_pin_blockages(2)

get_pin_buses

Creates a collection of pin_buses from the current design relative to the current instance. You can assign these pin_buses to a variable or pass them into another command.

SYNTAX

collection **get_pin_buses**
[-design *design*]
[-hierarchical]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-filter *expression*]
[-hsc *separator*]
patterns | -of_objects *objects*

Data Types

design collection
exact_count int
minimum_count int
expression string
patterns list
separator string
objects list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-hierarchical

Searches for pin_buses level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX **find** command.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-filter *expression*

Filters the collection with *expression*. For any pin_buses that match *patterns* (or *objects*) the expression is evaluated based on the pin_bus's attributes. If the expression evaluates to true, the pin_bus is included in the result.

-hsc *separator*

Specifies a separator character. The default is /. Valid values are /, @, ^, #, ., and |.

-of_objects *objects*

Creates a collection of pin_buses associated to the specified objects. In this case, each object in the list is a name, pattern, or collection. The objects can be a pin or cell_bus. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

patterns

Matches pin_bus names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type pin_bus. Supports patterns containing hierarchical separator when **-hierarchical** is specified.*patterns* and **-of_objects** are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_pin_buses** command creates a collection of pin_buses in the current design, relative to the current instance, that match certain criteria. The command returns a collection if any pin_buses match the *patterns* or *objects* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

A pin_bus represents an array of pins created with the same base name. This is typically accomplished in the source HDL using an

array notation.

If any *patterns* (or *objects*) fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_pin_buses** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_pin_buses** result to a variable.

When issued from the command prompt, **get_pin_buses** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_pin_buses** provides a fast, simple way to display pin_buses in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_pin_buses** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the pin_buses that begin with "reg_bus". Although the output looks like a list, it is not. The output is just a display.

```
prompt> get_pin_buses "*reg_bus*"
{"reg_cell_bus/reg_bus_x", "reg_cell_bus/reg_bus_y"}
```

SEE ALSO

collections(2)
create_pin_bus(2)
filter_collection(2)
get_pins(2)
link_block(2)
query_objects(2)
regexp(2)
remove_pin_buses(2)
report_pin_buses(2)
shell.common.collection_result_display_limit(3)

get_pin_constraints

Creates a collection of pin_constraints. You can assign these pin_constraints to a variable or pass them into another command.

SYNTAX

collection **get_pin_constraints**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
patterns | -of_objects *objects*

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns list
objects collection

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the search results to those matching the given expression. The filter expression must be a boolean expression based on the pin constraint attributes.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are

mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches pin_constraint names against patterns. This command uses a default pattern of "*" if the user does not specify any arguments. Mutually exclusive with the **-of_objects** argument.

-of_objects *objects*

Specifies a list of objects from which to find associated pin constraints. Supported object types are block, which returns pin constraints owned by the given block, and pin/port/net/cell/bundle, which return the pin constraints that apply to the given objects. Mutually exclusive with the **patterns** argument.

DESCRIPTION

The **get_pin_constraints** command creates a collection of pin_constraints in the current design, that match certain criteria. The command returns a collection if any pin_constraints match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

If any *patterns* fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_pin_constraints** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_pin_constraints** result to a variable.

When issued from the command prompt, **get_pin_constraints** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable **collection_result_display_limit**.

The "implicit query" property of **get_pin_constraints** provides a fast, simple way to display pin_constraints in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_pin_constraints** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

SEE ALSO

collections(2)
create_pin_constraint(2)
filter_collection(2)
remove_pin_constraints(2)
report_pin_constraints(2)

get_pin_guides

Creates a collection of pin guides from the current design.

SYNTAX

collection **get_pin_guides**
[-filter *expression*]
[-quiet]
[-design *design_name*]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

expression string
design_name string
exact_count int
minimum_count int
patterns list
objects list
region list
point list

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the pin guide attributes. You can determine the pin guide attributes by using the **list_attributes** command. If the expression evaluates to **true**, the pin guide is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-design *design*

Specifies the top design for finding objects. If this is not specified, objects are found in the current design.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for pin guides level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

patterns

Matches pin guide names against patterns. Patterns can include the wildcard characters the asterisk (*) and question mark (?) wildcard characters or regular expressions, depending on the use of the **-regexp** option. For more details about using and escaping wildcards, see the **wildcards** man page.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-of_objects *of_objects*

Creates a collection containing the pin guides associated with the specified objects. The objects can be pins, nets, or ports.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all pin_guides at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all pin_guides that are completely inside the specified region and doesn't include pin_guides which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all pin_guides that are completely inside the specified region and the pin_guides which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all pin_guides which are abut/touching from inside or outside to the specified region and the pin_guides whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of pin guides from the current design that match certain criteria. The command returns a collection handle (identifier) if any pin guides match the value of the *patterns* option and pass the filter criteria (if specified). If no objects match the criteria, the command returns an empty string.

You can use the **get_pin_guides** command at the command prompt or nest it as an argument to another command (such as **report_pin_guides**). You can also assign the **get_pin_guides** result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following returns the pin guides associated with the specified pins.

```
prompt> get_pin_guides -of_objects [get_pins cellA/pin*]
{VA1}
```

SEE ALSO

create_pin_guide(2)
remove_pin_guides(2)
report_pin_guides(2)
add_to_pin_guide(2)
remove_from_pin_guide(2)

get_pins

Creates a collection of pins from the netlist. You can assign the collection to a variable or pass the collection to another command.

SYNTAX

collection **get_pins**
[-hierarchical]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-leaf]
[-physical_context]
[-design *design_name*]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hsc *separator_character*]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

expression string
design_name string
exact_count integer
minimum_count integer
separator_character string
patterns list
objects list
region list
point list

ARGUMENTS

-hierarchical

Searches for pins level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX **find** command. For example, if there is a pin block1/adder/D[0], a hierarchical search

finds it using "adder/D[0]". This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-filter *expression*

Filters the collection with *expression*. For any pins that match *patterns* or *objects*, the expression is evaluated based on the attributes of the pin. If the expression evaluates to true, the pin is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as true regular expressions rather than as simple wildcard patterns. This option modifies the behavior of the =~ and !~ filter operators to compare with true regular expressions rather than simple wildcard patterns. The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Performs case-insensitive pattern matching.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The **-exact** and **-regexp** options are mutually exclusive.

-leaf

Includes only pins on leaf cells connected to the specified nets. You can use this option only with **-of_objects**. For any nets in the *objects* argument to **-of_objects**, only pins on leaf cells connected to those nets are included in the collection. In addition, hierarchical boundaries are crossed to find pins on leaf cells.

-physical_context

Searches only for physical pins. This option can also be used with **-of_objects**. For any nets in the *objects* argument to **-of_objects**, only pins on physical cells connected to those nets are included in the collection. In addition, hierarchical boundaries are crossed to find pins on physical cells. For any cells in the *objects* argument to **-of_objects**, pins of the cell are returned, if the cell is physical. If **-of_objects**, is not used than physical pins matching the specified *patterns* are returned.

-design *design_name*

Specifies the name of the design in which to find pins.

-expect *exact_count*

Specifies an exact number of objects to find. If the command finds a different number of objects, the command raises a Tcl error and stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, the command raises a Tcl error and stops.

-expect_each_pattern_matches

Ensures that each pattern in *patterns* matches at least one object. If one or more patterns does not match, the command raises a Tcl error and stops.

-hsc separator_character

Specifies a separator character. The default is /. Valid values are /, @, ^, #, ., and |.

patterns

Matches pin names against patterns. Patterns can include the wildcard characters "*" and "?", or regular expressions, based on the **-regexp** option. Patterns can also include collections of type pin. Patterns can also include subscript patterns, but not in conjunction with wildcard and regular expressions pattern matching. The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-of_objects ***objects***

Creates a collection of pins connected to the specified objects. Each object is a named cell, named net, named stub_chain, named scan_chain, cell collection, net collection, stub_chain collection, scan_chain collection, safety register rule, or safety register group

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at ***point***

Creates a collection that contains all pins at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within ***region***

Creates a collection that contains all pins that are completely inside the specified region and doesn't include pins which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching ***region***

Creates a collection that contains all pins that are completely inside the specified region and the pins which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one

except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect ***region***

Creates a collection that contains all pins which are abut/touching from inside or outside to the specified region and the pins whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

The **get_pins** command creates a collection of pins in the current design, relative to the current instance, that match certain criteria. The command returns a collection of pins that match the specified *patterns* or *objects*, and pass the filter, if specified. If no objects match the criteria, the empty string is returned.

When used with **-of_objects**, **get_pins** searches for pins connected to any cells or nets specified in *objects*. For net objects, there are two variations of pins that are considered. By default, only pins connected to the net at the same hierarchical level are considered. When combined with the **-leaf** option, only pins connected to the net that are on leaf cells are considered. In this case, hierarchical boundaries are crossed to find pins on leaf cells. Note that **-leaf** has no effect on the pins of cells.

If no *patterns* (or *objects*) match any objects, and the current design is not linked, the design automatically links.

When a cell has bus pins, **get_pins** can find them in several ways. For example, if cell u1 has a bus "A" with indexes 2 to 0, and the **design.bus_delimiters** application option is set to [], use "u1/A[*]" as the pattern. You can also find the same three pins with "u1/A" as the pattern.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

Subscript patterns are also supported, but cannot be used in conjunction with wildcard or regular expression pattern matching. Bus subscripting is specified with the format "name[begin_index:end_index:step_n]", and returns every nth bus pin inclusively between the begin and end indices. Note that the "[]" characters must be escaped, because they are Tcl special characters (e.g., A\[0:15:2\]). Name expansion subscripting is specified with the format "name(begin_index:end_index:step_n)", and returns every nth pin inclusively between the begin and end indices, which has a matching name without the delimiters characters (i.e., pin names do not actually contain the '(' and ')' characters). Note that ":step_n" is optional. The default step is 1. Multiple "begin_index:end_index:step_n" patterns can be specified, separated with a ',' character.

You can use the **get_pins** command at the command prompt, or you can nest it as an argument to another command (for example,

query_objects). In addition, you can assign the **get_pins** result to a variable.

When issued from the command prompt, **get_pins** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can increase or decrease the number of results that are displayed by setting the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_pins** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_pins** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the 'CP' pins of cells that begin with 'o'. Although the output looks like a list, it is just a display.

```
prompt> get_pins o*/CP
{"o_reg1/CP", "o_reg2/CP", "o_reg3/CP", "o_reg4/CP"}
```

The following example queries pins using bus subscripting.

```
prompt> get_pins {U1/A[0:1,3:7:2]}
{"U1/A[0]", "U1/A[1]", "U1/A[3]", "U1/A[5]", "U1/A[7]"}
```

The following example queries pins using name expansion subscripting.

```
prompt> get_pins U1/B(0:4:2)C(1:2,5)
{"U1/B0C1", "U1/B0C2", "U1/B0C5", "U1/B2C1", "U1/B2C2", "U1/B2C5",
 "U1/B4C1", "U1/B4C2", "U1/B4C5"}
```

The following example queries the pins connected to the specified cells.

```
prompt> set csel [get_cells o_reg1]
{"o_reg1"}
prompt> query_objects [get_pins -of_objects $csel]
{"o_reg1/D", "o_reg1/CP", "o_reg1/CD", "o_reg1/Q", "o_reg1/QN"}
```

The following example shows the difference between getting local pins of a net and getting the leaf pins of net. In this example, NET1 is connected to i2/a and reg1/QN. Cell i2 is hierarchical. Within i2, port a is connected to the U1/A and U2/A.

```
prompt> get_pins -of_objects [get_nets NET1]
{"i2/a", "reg1/QN"}
prompt> get_pins -leaf -of_objects [get_nets NET1]
{"i2/U1/A", "i2/U2/A", "reg1/QN"}
```

The following example shows how to create a clock using a collection of pins.

```
prompt> create_clock -period 8 -name CLK [get_pins o_reg*/CP]
1
```

SEE ALSO

collections(2)
create_clock(2)
filter_collection(2)
get_cells(2)
link_block(2)
query_objects(2)
regexp(2)
design.bus_delimiters(3)
shell.common.collection_result_display_limit(3)

get_placement_attractions

Find and return existing placement_attraction objects

SYNTAX

collection **get_placement_attractions**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count int
minimum_count int
of_objects list
patterns list
region list
point list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the placement_attraction attributes. You can determine the placement_attraction attributes by using the **list_attributes** command. If the expression

evaluates to **true**, the placement_attraction is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for placement_attractions level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect**, and **-at**, but not with **-of_objects**.

-of_objects *of_objects*

Creates a collection containing the placement_attractions of the specified cells, ports, or placement_attraction shapes. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

patterns

Matches the placement_attraction names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-at *point*

Creates a collection that contains all placement_attractions at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all placement_attractions that are completely inside the specified region and doesn't include placement_attractions which are abut/touching from inside with the specified region. The region placement_attractionary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all placement_attractions that are completely inside the specified region and the placement_attractions which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all placement_attractions which are abut/touching from inside or outside to the specified region and the placement_attractions whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. If

you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of placement_attractions that meet the selection criteria. It returns a collection handle if one or more placement_attractions meet the selection criteria. If no attractions match the selection criteria, it returns an empty string.

You can use the **get_placement_attractions** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of placement_attractions by name pattern matching.

```
prompt> get_placement_attractions attract*
{"attract1", "attract2", "attract3"}
```

The following example creates a collection of placement_attractions with attribute filtering.

```
prompt> get_placement_attractions -filter {effort == medium}
{"attact1"}
```

The following example creates a collection of all placement_attractions.

```
prompt> get_placement_attaractions *
{"attract1", "attract2", "attract3", "attactA"}
```

The following example creates a collection of all placement_attaractions containing the cells "inv*" and port "reset".

```
prompt> get_placement_attaractions -of_objects "inv* reset"}
{"attract1", "attract2"}
```

SEE ALSO

add_to_placement_attraction(2)
collections(2)
create_placement_attraction(2)
filter_collection(2)
query_objects(2)
regexp(2)
remove_placement_attraction(2)
remove_from_placement_attraction(2)
report_placement_attractions(2)

shell.common.collection_result_display_limit(3)

get_placement_blockages

Creates a collection of placement blockages from the current design.

SYNTAX

collection **get_placement_blockages**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count integer
minimum_count integer
patterns list
point list
region list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with expression. For any placement blockages that match the *patterns* argument, the expression is evaluated based on the attributes of the placement blockages. If the expression evaluates to *true*, the placement blockage is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for placement blockages level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**.

patterns

Specifies the placement blockages. The patterns can be a collection handle of blockages or other formats, similar to the following:

- * (asterisk) indicates all placement blockages
- *xx** indicates any placement blockage whose name begins with *xx*
- PB_77 indicates one placement blockage that does not yet have a name and its object_id is 77
- PB_* indicates any placement blockage that does not yet have a name.

The *patterns*, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all placement_blockages at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the

pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all placement_blockages that are completely inside the specified region and doesn't include placement_blockages which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all placement_blockages that are completely inside the specified region and the placement_blockages which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all placement_blockages which are abut/touching from inside or outside to the specified region and the placement_blockages whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical

hierarchies.

DESCRIPTION

This command creates a collection of placement blockages that meet the selection criteria. It returns a collection of placement blockages if one or more blockages meet the selection criteria. If no blockages match the selection criteria, it returns an empty string. You can use the **get_placement_blockages** command as an argument to another command or assign its result to a variable. Refer to the example below for details.

You can use the **get_placement_blockages** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** man page for information about working with collections.

EXAMPLES

The following examples show some uses of this command to create placement blockages:

```
prompt> get_placement_blockages *
{"PB_1", "PB_2", "PB_3"}

prompt> get_placement_blockages * -filter "area > 1000"
{"PB_3"}
```

SEE ALSO

collections(2)
create_placement_blockage(2)
get_attribute(2)
query_objects(2)

get_placement_ir_drop_target

Get placer IR drop targets

SYNTAX

string **get_placement_ir_drop_target** *voltage_area*

list *voltage_area*

ARGUMENTS

voltage_area

Specifies the voltage area for settings the IR drop targets

DESCRIPTION

This command returns the IR aware placement target voltage drops for the given voltage areas. The result is a list of IR drop targets, each IR drop target itself is a list of three or four items: the voltage area name; "low" or "high" to denote the voltage drop target type, the voltage drop target (in percentage) and optionally the word "-irdrop" if the target number is not a populastion percentage but a voltage drop percentage.

EXAMPLES

The following example shows the use of **get_placement_ir_drop_target** command:

```
prompt> set_placement_ir_drop_target DEFAULT_VA low 1.0 -irdrop
prompt> set_placement_ir_drop_target DEFAULT_VA high 2.0
prompt> get_placement_ir_drop_target DEFAULT_VA
{ { DEFAULT_VA low 1.000000 -irdrop } { DEFAULT_VA high 2.000000 } }
```

SEE ALSO

set_placement_ir_drop_target(2)
reset_placement_ir_drop_target(2)
report_placement_ir_drop_target(2)
place.coarse.ir_drop_default_target_low_population(3)
place.coarse.ir_drop_default_target_low_percentage(3)
place.coarse.ir_drop_default_target_high_population(3)
place.coarse.ir_drop_default_target_high_percentage(3)

get_port

Creates a collection of ports from the current design. You can assign these ports to a variable or pass them into another command.

SYNTAX

collection **get_ports** [-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-physical_context]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

expression string
exact_count integer
minimum_count integer
patterns list
objects list
point list
region list

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any ports that match *patterns*, the expression is evaluated based on the port's attributes. If the expression evaluates to true, the cell is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the

=~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-of_objects ***objects***

Creates a collection of ports connected to or associated with the specified objects. Each object is a named cell, cell collection, named net, net collection, named port bus, port bus collection, named stub_chain, stub_chain collection, named scan_chain or scan_chain collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-physical_context

This option is needed when a cell object is provided as input to **-of_objects** option. The ports of the cells are returned, if the cell is a physical hierarchical cell.

-expect ***exact_count***

Specifies an exact number of objects to find. If the command finds a different number of objects, the command raises a Tcl error and stops.

-expect_at_least ***minimum_count***

Specifies a minimum number of objects to find. If the command finds fewer objects, the command raises a Tcl error and stops.

-expect_each_pattern_matches

Ensures that each pattern in *patterns* matches at least one object. If one or more patterns does not match, the command raises a Tcl error and stops.

patterns

Matches port names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type port. Patterns can also include subscript patterns, but not in conjunction with wildcard and regular expressions pattern matching.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at ***point***

Creates a collection that contains all ports at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-within ***region***

Creates a collection that contains all ports that are completely inside the specified region and doesn't include ports which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-touching *region*

Creates a collection that contains all ports that are completely inside the specified region and the ports which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

In addition, you cannot use the **-hierarchical** option with the **-touching** option.

-intersect *region*

Creates a collection that contains all ports which are abut/touching from inside or outside to the specified region and the ports whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

DESCRIPTION

The **get_ports** command creates a collection of ports in the current design that match certain criteria. The command returns a collection if any ports match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

Subscript patterns are also supported, but cannot be used in conjunction with wildcard or regular expression pattern matching. Bus subscripting is specified with the format "name[begin_index:end_index:step_n]", and returns every nth bus port inclusively between the begin and end indices. Note that the "[]" characters must be escaped, because they are Tcl special characters (e.g., A\[0:15:2\]). Name expansion subscripting is specified with the format "name(begin_index:end_index:step_n)", and returns every nth port inclusively between the begin and end indices, which has a matching name without the delimiters characters (i.e., port names do not actually contain the '(' and ')' characters). Note that ":step_n" is optional. The default step is 1. Multiple "begin_index:end_index:step_n" patterns may be specified, separated with a ',' character.

You can use the **get_ports** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_ports** result to a variable.

When issued from the command prompt, **get_ports** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the application option **shell.common.collection_result_display_limit**.

The "implicit query" property of **get_ports** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_ports** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page. In addition, refer to the man pages for **all_inputs** and **all_outputs**, which also create collections of ports.

EXAMPLES

The following example queries all input ports beginning with 'mode'. Although the output looks like a list, it is just a display.

```
prompt> get_ports "mode*" -filter {direction == in}
{"mode[0]", "mode[1]", "mode[2]"}
```

The following example queries ports using bus subscripting.

```
prompt> get_ports {A[0:1,3:7:2]}
{"A[0]", "A[1]", "A[3]", "A[5]", "A[7]"}
```

The following example queries ports using name expansion subscripting.

```
prompt> get_ports B(0:4:2)C(1:2,5)
{"B0C1", "B0C2", "B0C5", "B2C1", "B2C2", "B2C5",
 "B4C1", "B4C2", "B4C5"}
```

The following example sets the driving cell for ports beginning with "in" to an FD2.

```
prompt> set_driving_cell -cell FD2 -library my_lib [get_ports in*]
```

The following example reports ports connected to nets matching the pattern "bidir*".

```
prompt> report_port [get_ports -of_objects [get_nets "bidir*"]]
```

SEE ALSO

all_inputs(2)
all_outputs(2)
collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_port_antenna_property

Get antenna attribute values on the specified port.

SYNTAX

antenna_prop_list **get_port_antenna_property**
-port *port*
[-layer *pin_layer*]

antenna_prop_list *collection*

Data Types

port blk port or lib-pin
layer layer object

ARGUMENTS

-port *port*

Specify the port or lib_pin from which the antenna property is to be retrieved.

This option is required.

-layer *search_layer*

Specifies the layer only for which the antenna properties for the port will be returned.

If you do not specify this option, the tool will return all the antenna properties on all the layers. If you specify this option, only those antenna properties on the specified layer will be returned.

DESCRIPTION

This command gets antenna property data from the specified port or lib_pin. It returns the data as a Tcl array of the form:

{[well_type] [oxide_model <type>] metalLayer metalLayer1 gate_size gate_data
mode1_area mode1_data mode2_ratio mode2_data mode3_area mode3_data
mode4_area mode4_data mode5_ratio mode5_data mode6_area mode6_data}

In the event no layer is given, the data for all metal layers that have antenna data is returned in the following form (abbreviated here for clarity):

```
{
 {[well_type] [oxide_model <type>] metalLayer metalLayer1 gate_size gate_data1 ...}
 {[well_type] [oxide_model <type>] metalLayer metalLayer2 gate_size gate_data1 ...}
 ...
}
```

Note that the returned data is a Tcl array and as such can be easily modified.

RETURN VALUE

antenna_prop_list if successful, empty list otherwise

EXAMPLES

The following examples depict the behavior.

```
prompt> get_port_antenna_property -port p3 -layer metal3
{{metalLayer metal3 gate_size 0.3436 mode1_area 6.664
  mode2_ratio 2.05346 mode3_area 12.708 mode4_area 17.5392
  mode5_ratio 7.04044 mode6_area 12.682} {well1 metalLayer metal3
  gate_size 0.8436 mode1_area 7.164 mode2_ratio 2.55346 mode3_area
  13.208 mode4_area 18.0392 mode5_ratio 7.54044 mode6_area 13.182}}

prompt> get_port_antenna_property -port p3 -layer metal3
{}
```

SEE ALSO

derive_hier_antenna_property(2)
set_port_antenna_property(2)

get_port_buses

Creates a collection of port buses from the current design. You can assign these port buses to a variable or pass them into another command.

SYNTAX

collection **get_port_buses**
[-design *design*]
[-physical_context]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
patterns | -of_objects *objects*

Data Types

design string or collection
expression string
exact_count integer
minimum_count integer
patterns list
objects string or collection

ARGUMENTS

-design *design_name*

Specifies the top-level design for finding objects. If this is not specified, objects are found in the current design.

-physical_context

This option is needed when a cell object is provided as input to the **-of_objects** option. The port buses of the cells are returned, if the cell is a physical hierarchical cell.

-filter *expression*

Filters the collection with *expression*. For any port buses that match *patterns*, the expression is evaluated based on the port's attributes. If the expression evaluates to true, the cell is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Specifies that the *patterns* argument is a real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. Use this option when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-of_objects *objects*

Creates a collection of port buses connected to the specified objects. Each object is a named cell, cell collection, named port, port collection, named block or block collection. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

patterns

Matches port names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type port. Patterns can also include subscript patterns, but not in conjunction with wildcard and regular expressions pattern matching. *patterns* and **-of_objects** are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_port_buses** command creates a collection of port buses in the current design that match the specified criteria. The command returns a collection if any port buses match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the **regexp** Tcl command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_port_buses** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_port_buses** result to a variable.

When issued from the command prompt, **get_port_buses** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_port_buses** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_port_buses** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page. In addition, refer to the man pages for **all_inputs** and **all_outputs**, which also create collections of port buses.

EXAMPLES

The following example queries all input port buses beginning with 'bus'. Although the output looks like a list, it is just a display.

```
prompt> get_port_buses "bus*"
{"bus1", "bus2"}
```

SEE ALSO

all_inputs(2)
all_outputs(2)
collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_ports

Creates a collection of ports from the current design. You can assign these ports to a variable or pass them into another command.

SYNTAX

collection **get_ports** [-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-physical_context]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

expression string
exact_count integer
minimum_count integer
patterns list
objects list
point list
region list

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any ports that match *patterns*, the expression is evaluated based on the port's attributes. If the expression evaluates to true, the cell is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the

=~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-of_objects ***objects***

Creates a collection of ports connected to or associated with the specified objects. Each object is a named cell, cell collection, named net, net collection, named port bus, port bus collection, named stub_chain, stub_chain collection, named scan_chain or scan_chain collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-physical_context

This option is needed when a cell object is provided as input to **-of_objects** option. The ports of the cells are returned, if the cell is a physical hierarchical cell.

-expect ***exact_count***

Specifies an exact number of objects to find. If the command finds a different number of objects, the command raises a Tcl error and stops.

-expect_at_least ***minimum_count***

Specifies a minimum number of objects to find. If the command finds fewer objects, the command raises a Tcl error and stops.

-expect_each_pattern_matches

Ensures that each pattern in *patterns* matches at least one object. If one or more patterns does not match, the command raises a Tcl error and stops.

patterns

Matches port names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type port. Patterns can also include subscript patterns, but not in conjunction with wildcard and regular expressions pattern matching.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at ***point***

Creates a collection that contains all ports at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-within ***region***

Creates a collection that contains all ports that are completely inside the specified region and doesn't include ports which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-touching ***region***

Creates a collection that contains all ports that are completely inside the specified region and the ports which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

In addition, you cannot use the **-hierarchical** option with the **-touching** option.

-intersect ***region***

Creates a collection that contains all ports which are abut/touching from inside or outside to the specified region and the ports whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

DESCRIPTION

The **get_ports** command creates a collection of ports in the current design that match certain criteria. The command returns a collection if any ports match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

Subscript patterns are also supported, but cannot be used in conjunction with wildcard or regular expression pattern matching. Bus subscripting is specified with the format "name[begin_index:end_index:step_n]", and returns every nth bus port inclusively between the begin and end indices. Note that the "[]" characters must be escaped, because they are Tcl special characters (e.g., A\[0:15:2\]). Name expansion subscripting is specified with the format "name(begin_index:end_index:step_n)", and returns every nth port inclusively between the begin and end indices, which has a matching name without the delimiters characters (i.e., port names do not actually contain the '(' and ')' characters). Note that ":step_n" is optional. The default step is 1. Multiple "begin_index:end_index:step_n" patterns may be specified, separated with a ',' character.

You can use the **get_ports** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_ports** result to a variable.

When issued from the command prompt, **get_ports** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the application option **shell.common.collection_result_display_limit**.

The "implicit query" property of **get_ports** provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_ports** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page. In addition, refer to the man pages for **all_inputs** and **all_outputs**, which also create collections of ports.

EXAMPLES

The following example queries all input ports beginning with 'mode'. Although the output looks like a list, it is just a display.

```
prompt> get_ports "mode*" -filter {direction == in}
{"mode[0]", "mode[1]", "mode[2]"}
```

The following example queries ports using bus subscripting.

```
prompt> get_ports {A[0:1,3:7:2]}
{"A[0]", "A[1]", "A[3]", "A[5]", "A[7]"}
```

The following example queries ports using name expansion subscripting.

```
prompt> get_ports B(0:4:2)C(1:2,5)
{"B0C1", "B0C2", "B0C5", "B2C1", "B2C2", "B2C5",
 "B4C1", "B4C2", "B4C5"}
```

The following example sets the driving cell for ports beginning with "in" to an FD2.

```
prompt> set_driving_cell -cell FD2 -library my_lib [get_ports in*]
```

The following example reports ports connected to nets matching the pattern "bidir*".

```
prompt> report_port [get_ports -of_objects [get_nets "bidir*"]]
```

SEE ALSO

all_inputs(2)
all_outputs(2)
collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_power_budget

Returns the power budget for either the current design or a list of cells in the current design.

SYNTAX

status **get_power_budget**
[-scenarios *scenario_list*]
[-cell *cell_list*]

Data Types

scenario_list list
cell_list list

ARGUMENTS

-scenarios *scenario_list*

Indicates the scenario for which the budget values are returned. If no scenario is specified the current_scenario is used.

-cell *cell_list*

Specifies a list of cells on which the power budget values are reported.

DESCRIPTION

This command is used to return the budget values either on the design or on specific cells in the design. If this command is called without any option, the power budget value for the current design is returned.

If the command is called with the -cell option, then budget values will only be reported for the instances specified in the cell list.

Returns the requested numbers as a tagged list.

Multicorner-Multimode Support

EXAMPLES

In the following example, power budget value has been set globally on the design.

```
prompt> set_power_budget 5.92e+05
prompt> get_power_budget

{{Scenario SC1} {Design Top} {Power_Budget 5.92e+05}}
```

In the following example power budget has been set globally on the design, as well as on the hierarchical cell H1/H2.

```
prompt> set_power_budget 3.9e+05
prompt> set_power_budget -cell [get_cell H1/H2] 2.3e+03
prompt> get_power_budget [get_cells H1/H2]

{{Scenario SC1} {Cell H1/H2} {Power_Budget 2.3e+03}}
```

SEE ALSO

get_power_budget(2)
reset_power_budget(2)
report_power_budget(2)
report_power(2)

get_power_clock_scaling

Returns the clock frequency scaling values stored.

SYNTAX

status **get_power_clock_scaling**
[-scenarios *scenarios*]
[clock_list]

Data Types

scenarios list
clock_list list

ARGUMENTS

-scenarios *scenarios*

Specify a list of scenarios for which the scaling factor is to be retrieved, if found. If not specified, then scaling data is filtered for only the clock objs specified through clock_list option. If both scenarios and clocks are not specified then complete scaling data is printed. If the app option power.get_power_clock_scaling_single_objects_only is set to 'true', this option can take atmost one value. If the app option is set to true, and if this option is not used, the current scenario is assumed.

clock_list

Specify a list of clocks in the design for which the scaling factor is to be retrieved, if found. If not specified, then scaling data is filtered for only the scenarios specified through -scenarios option. If both scenarios and clocks are not specified then complete scaling data is printed. If the app option power.get_power_clock_scaling_single_objects_only is set to 'true', this option can take atmost one value. If the app option is set to true, and if this option is not used, the non-clock specific ratio value will be returned for the scenario.

DESCRIPTION

Clock frequency used in Switching Activity Interchange Format (SAIF) files, from logic simulation for functional verification can differ from the clock frequency used in Synopsys Design Constraints (SDC) file for timing and power analysis. This command retrieves the scaling factors stored in the set_power_clock_scaling command.

The clock_objects option in this command is a list of clocks in the design that have the specified period or ratio values used in simulation for SAIF generation. The scenarios are the ones for which the scaling factors are to be retrieved and displayed. Note that if both the scenario and the clock options are not specifed then the complete scaling data for all the scenarios is displayed.

When period value is specified for a clock, then the scaling ratio is calculated as the ratio of (period of the clock from set_power_clock_scaling cmd) / (period of the fastest clock of an object).This ratio is applied to the retrieved simulated activity of the object. If the ratio value is specified then it is directly applied as the scaling ratio.This ratio is applied to the retrieved simulated activity of the object.

If the app option power.get_power_clock_scaling_single_objects_only is set to 'true', this command will return only a tuple containing the scaling value. The value returned will be clock specific value if clock is specified and will be global ratio value if clock is not specified. If the scenario option is not specified & the app option is true, the current scenario is assumed for fetching the tuple.

EXAMPLES

The following examples describe the behaviour of the cmd.

```
prompt> set_power_clock_scaling {CLK CLK2 C_A} -period 2.6
prompt> set_power_clock_scaling -ratio 1
prompt> get_power_clock_scaling
------------------------------------------------------------
      Clock Scaling Data
------------------------------------------------------------
Scenario:default, Clock name:CLK, Period:2.600000
Scenario:default, Clock name:CLK2, Period:2.600000
Scenario:default, Clock name:C_A, Period:2.600000
------------------------------------------------------------
Scaling Ratios for Scenarios without Clocks
------------------------------------------------------------
Scenario:default, Ratio:1.000000
```

SEE ALSO

set_power_clock_scaling(2)
report_power_clock_scaling(2)
power.get_power_clock_scaling_single_objects_only(3)

get_power_derate

Returns the power derating factors for either the current design, a list of cells, library cells or all cells in a power group in the current design.

SYNTAX

status **get_power_derate**
[-scenarios *scenario_list*]
[-leakage]
[-switching]
[-internal]
[-user]
[*object_list*]

Data Types

scenario_list list
object_list list

ARGUMENTS

-scenarios *scenario_list*

Indicates the scenario for which the derate_value is applied. If no scenario is specified the current_scenario is used.

-leakage

Returns the leakage specific derating values only.

-switching

Returns the switching specific derating values only.

-internal

Returns the internal specific derating values only.

-user

Returns the actually stored power derating attributes if and only if they exist.

object_list

Specifies current a list of cells or library cells on which the power derating factors are reported.

DESCRIPTION

This command is used to return the derate factors either on the design, specific cells or library cells contained in the design. If this command is called without any option the current design power derating factors are returned.

If this command is called with option -user it will return the actually stored attributes if and only if they exist. Without -user, the command will return the used (inherited) derating factors. If the command is called with the object_list specified then derating factors will only be issued for the instances specified in the object list.

Returns requested numbers as a tagged list. If none of -leakage, -internal or -switching is specified it returns all three.

Multicorner-Multimode Support

EXAMPLES

In the following example, power derating factors have been set globally on the design.

```
prompt> set_power_derate -switching 0.92
prompt> set_power_derate -internal 1.15
prompt> set_power_derate -leakage 0.75
prompt> get_power_derate

{ {scenario default} {object Top} {leakage 0.75} {switching 0.92 } {internal 1.15 } }
```

In the following example power derating factors have been set globally on the design. More restrictive derates have been set on the hierarchical cell H1.

```
prompt> set_power_derate 0.95
prompt> set_power_derate -switching 1.06
prompt> set_power_derate -internal -leakage 0.82 [get_cell H1]
prompt> get_power_derate [get_cells H1/U1]

{ {scenario default} {object H1/U1} {leakage 0.82 } {switching 1.06} {internal 0.82 } }
```

In the following example only the power derating factors set specifically by the user are returned for the hierarchical cell H1.

```
prompt> get_power_derate -user [get_cells H1/U1]

{ {scenario default} {object H1/U1} {leakage 0.82 } {internal 0.82 } }
```

SEE ALSO

get_power_derate(2)
reset_power_derate(2)
report_power_derate(2)

get_power_domains

Creates a collection of power domains that meet the specified criteria.

SYNTAX

collection **get_power_domains**
[-hierarchical]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-of_objects *objects*]
[*patterns*]

Data Types

expression string
exact_count int
minimum_count int
objects list
patterns list

ARGUMENTS

-hierarchical

Searches for power domains in the current scope and all of its descendant scopes. If you specify this option, you can specify only simple names in the *patterns* argument; the name cannot contain hierarchy delimiter characters.

If you do not specify this option, the tool searches for power domains only in the current scope and you can use hierarchy delimiter characters in the *patterns* argument to specify hierarchical power domain names relative to the current scope.

You cannot specify this option when you use the **-of_objects** option.

-filter *expression*

Filters the collection with *expression*. For any power domains that match *patterns*, the expression is evaluated based on the power domain's attributes. If the expression evaluates to true, the power domain is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *objects*

Returns a collection of power domains associated with the specified objects. The objects can be cells, power strategies, voltage areas, or voltage area shapes.

The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either, the tool uses a pattern of "*".

patterns

Matches power domain names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, refer to the **wildcards** man page.

The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either, the tool uses "*".

DESCRIPTION

The **get_power_domains** command creates a collection of power domains in the current design that match certain criteria. If no power domains match the criteria, an empty string is returned.

Until the **commit_upf** command is executed and power intent is finalized, power domains of subblocks are not propagated and will not be included in the result.

Regular expression matching is the same as in the Tcl regexp command. When using **-regexp**, take care in the way you quote the patterns and filter expression. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_power_domains** command at the command prompt, or you can nest it as an argument to another command, such as **report_power_domains**. In addition, you can assign the **get_power_domains** result to a variable.

When issued from the command prompt, **get_power_domains** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum by using the **shell.common.collection_result_display_limit** application option.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example creates power domains, then queries them.

```
prompt> create_power_domain TOP_DOMAIN
1
prompt> create_power_domain SUB_DOMAIN -elements [get_cells mid1]
1
prompt> get_power_domains
{TOP_DOMAIN SUB_DOMAIN}
prompt> get_power_domains TOP*
{TOP_DOMAIN}
```

The following example gets all power domains under the current scope.

```
prompt> get_power_domains -hierarchical
{PD1 SUB_INST/PD2}
```

SEE ALSO

create_power_domain(2)
report_power_domains(2)

get_power_group

Returns list of all power groups the specified cell or libcell is in.

SYNTAX

status **get_power_group**
object
[-user]
[-default]

Data Types

object object

ARGUMENTS

-user

Returns user specified power groups for the specified *object* if exist, else an error message is reported stating no user specified power group found on that object.

-default

Returns default power group for the specified *object*.

object

Specifies cell or libcell for which power group needs to be reported.

DESCRIPTION

This command returns list of all power groups on the specified object. If '-user' is specified then user specified power groups are reported. If '-default' is specified then default power group is reported. If neither '-user' nor '-default' is specified then return value depends on the app option 'power.report_user_power_groups'. If the app option is set as 'exclusive' then command returns Actual power group. Actual power group is the user specified power group present for that cell. If multiple user power groups are specified then first power group in the specified list is Actual power group. If user specified power group is not present for that cell then actual power group is the user specified power group present for the libcell of that cell. If multiple user power groups are present for the libcell then first power group in the specified list is Actual power group. If user specified power group is not present for that libcell then actual power group is the default power group of that cell. If app option value is 'inclusive' then list of all user specified power groups and default power group of that cell is returned. If cell has no user power groups specified then the corresponding libcell is checked for user specified power groups. It is to be noted that '-user' and '-default' options can not be specified together.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example returns default power group of a cell.

prompt> **get_power_group u3/S_reg_0 -default**

SEE ALSO

set_power_group(2)
reset_power_group(2)
report_power_groups(2)
get_power_group_objects(2)
report_power(2)
power.report_user_power_groups(3)

get_power_group_objects

Returns a collection of cells for the specified power groups.

SYNTAX

status **get_power_group_objects**
group_names

Data Types

group_names list

ARGUMENTS

group_names

Specifies the power groups for which the collection of cells will be returned.

DESCRIPTION

The get_power_group_objects command returns a collection of cells contained by the specified power groups. The return value depends on app option 'power.report_user_power_groups'. The collection will have no duplicate objects.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example returns collection of cells present in power group macro1 and register.

prompt> **get_power_group_objects {macro1 register}**

SEE ALSO

set_power_group(2)
reset_power_group(2)
get_power_group(2)
report_power_groups(2)
report_power(2)
power.report_user_power_groups(3)

get_power_strategies

Creates a collection of power strategies.

SYNTAX

collection **get_power_strategies**
[-domain *power_domain*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-of_objects *objects*]
[*patterns*]

Data Types

power_domain string
expression string
exact_count integer
minimum_count integer
patterns list

ARGUMENTS

-domain *power_domain*

Specifies the name of the power domain in which to collect this UPF power strategy.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the power strategies attributes. You can determine the power strategies attributes by using the **list_attributes** command. If the expression evaluates to **true**, the power strategy is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~

and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Specifies that each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-of_objects *objects*

Returns a collection of power strategies associated with the specified objects. The objects can be cells or power domains.

patterns

Matches the power strategy names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns* argument is not specified, the command uses asterisk (*) as the *pattern*.

DESCRIPTION

This command creates a collection of power strategies that match certain criteria. The command returns a collection handle (identifier) if any power strategies match the value of the *patterns* option and pass the filter criteria (if specified). If no objects match the criteria, the command returns an empty string.

Until the **commit_upf** command is executed and power intent is finalized, strategies of subblocks are not propagated and are not included in the result.

You can use the **get_power_strategies** command at the command prompt or nest it as an argument to another command (such as **get_cells**, **set_power_strategy**). You can also assign the **get_power_strategies** result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following returns isolation strategies defined in specified power domain.

```
prompt> get_power_strategies -domain TOP -filter {type==ISO}
{iso_stra1 iso_stra2}
```

SEE ALSO

create_power_switch(2)
set_isolation(2)
set_level_shifter(2)
set_power_strategy_attribute(2)
set_retention(2)

get_power_switch_patterns

Creates a collection of power switch patterns that meet the specified criteria.

SYNTAX

collection **get_power_switch_patterns**
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns*]

Data Types

expression string
exact_count int
minimum_count int
objects list
patterns list

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For any power switch patterns that match *patterns*, the expression is evaluated based on the attributes of the power switch pattern. If the expression evaluates to true, the power switch pattern is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches power switch pattern names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, refer to the **wildcards** man page.

DESCRIPTION

The **get_power_switch_patterns** command creates a collection of power switch patterns in the current design, that match certain criteria. If no power switch pattern match the criteria, an empty string is returned.

Regular expression matching is the same as in the Tcl regexp command. When using **-regexp**, take care in the way you quote the patterns and filter expression. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_power_switch_patterns** command at the command prompt, or you can nest it as an argument to another command, such as **report_power_switch_patterns**. In addition, you can assign the **get_power_switch_patterns** result to a variable.

When issued from the command prompt, **get_power_switch_patterns** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example gets all power switch patterns in the current design.

```
prompt> get_power_switch_patterns
{PSW_PATTERN_INST_pattern2_0 PSW_PATTERN_INST_pattern2_1 PSW_PATTERN_INST_pattern2_2}
```

The following example gets all power switch patterns matching a given pattern.

```
prompt> get_power_switch_patterns *pattern2_7*
{PSW_PATTERN_INST_pattern2_7 PSW_PATTERN_INST_pattern2_70 PSW_PATTERN_INST_pattern2_71}
```

SEE ALSO

report_power_switch_patterns(2)

get_pr_rules

Returns a collection of cell row spacing rules (pr_rules) from the specified technology object.

SYNTAX

collection **get_pr_rules**
[-tech *tech_object*]
[-library *library*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-of_objects *of_objects*]
[*patterns*]

Data Types

tech_object collection
library collection
expression string
exact_count int
minimum_count int
of_objects list
patterns string

ARGUMENTS

-tech *tech_object*

Specifies the technology object in which to search for the cell row spacing rules.

The **-tech** and **-library** options are mutually exclusive; you can specify only one. If you do not specify either option, the rule is searched in the technology object of the current library.

-library *library*

Specifies the library in which to search for the cell row spacing rules. The command searches for the rules in the technology object associated with the specified library. In an implementation tool, the library is a design library. In the library manager, the library is a cell library. You can specify the library using the library's name or a collection that contains the library.

The **-tech** and **-library** options are mutually exclusive; you can specify only one. If you do not specify either option, the rule is

searched in the technology object of the current library.

-filter *expression*

Filters the collection with *expression*. For any rules that match the *patterns* or *objects* option, the expression is evaluated based on the attributes of the rule. If the expression evaluates to true, the rule is included in the result.

-quiet

Suppresses warning and error messages if no object matches. Syntax error messages are not suppressed.

-regexp

Considers the *patterns* argument as true regular expressions rather than simple wildcard patterns. This option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

The **-regexp** and **-exact** options are mutually exclusive; you can specify only one.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

The **-regexp** and **-exact** options are mutually exclusive; you can specify only one.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, it raises a Tcl error and stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, it raises a Tcl error and stops.

-expect_each_pattern_matches

Requires each pattern in the *patterns* argument to match at least one object. If one or more patterns do not match, the command raises a Tcl error and stops.

-of_objects *of_objects*

Creates a collection that contains the cell row spacing rules of the specified technology objects. Each object in the list is a name, pattern, or collection.

The **-of_objects** option and *patterns* argument are mutually exclusive; you can specify only one. If you do not specify either, the wildcard "*" is used as the *pattern*.

patterns

Matches the rule names against the *patterns* argument. The *patterns* argument can include the "*" wildcard character.

The **-of_objects** option and *patterns* argument are mutually exclusive; you can specify only one. If you do not specify either, the wildcard "*" is used as the *pattern*.

DESCRIPTION

The **get_pr_rules** command creates a collection of cell row spacing rules from the specified technology object that match certain

criteria. The rules include both the rules specified in the PRRule section of the technology file and the rules created with the **create_pr_rule** command.

The command returns a collection of rules if at least one cell row spacing rule matches the criteria; otherwise, it returns an empty string.

You can use the **get_pr_rules** command at the command prompt or nest it as an argument to another command, such as **report_attributes**. In addition, you can assign the **get_pr_rules** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a collection that contains all the cell row spacing rules in the technology object of the current library.

```
prompt> get_pr_rules
{PR_RULE_0}
```

SEE ALSO

create_pr_rule(2)
remove_pr_rules(2)
report_pr_rules(2)

get_purposes

Creates a collection of one or more purposes.

SYNTAX

collection **get_purposes**
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *of_objects*]

Data Types

expression string
exact_count int
minimum_count int
of_objects list
patterns string

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*. For any purposes that match, the expression is evaluated based on the layer's attributes. If the expression evaluates to true, the layer is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect ***exact_count***

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least ***minimum_count***

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects ***of_objects***

Creates a collection containing the purposes of the specified libs. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches layer names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark) or regular expressions, depending on whether or not you also use the **-regexp** option.

DESCRIPTION

This command creates a collection of purposes defined in the technology file for the current library.

You can use the **get_purposes** command at the command prompt or as an argument nested in another command (for example, in the **query_objects** command). You can also assign its result to a variable.

The layer name printed for returned collection will have following format.

layer_name:purpose_name

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a layer for the name M1R.

```
prompt> get_purposes M1R
{drawing}
```

SEE ALSO

collections(2)
filter_collection(2)
query_objects(2)
create_purpose(2)
shell.common.collection_result_display_limit(3)

get_related_supply_nets

Creates a collection of related supply nets of signal pins or ports.

SYNTAX

collection **get_related_supply_nets**
[-filter *expression*]
[-quiet]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expand]
[-ground]
[-in_block_upf]
objects

Data Types

expression string
exact_count int
minimum_count int
objects list

ARGUMENTS

-filter *expression*

Filters the collection with *expression*. For each related supply net, the expression is evaluated based on the supply net's attributes. If the expression evaluates to true, the supply net is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expand

Do not merge and combine duplicated related supply nets of signal pins.

-ground

Specifies that the ground supply net should be returned.

-in_block_upf

This is for boundary ports of linked blocks only. Return the related supply net in block UPF. The return value shows the related supply net for this block boundary port when the linked block is opened as current block.

objects

Returns a collection of related supply nets associated with the signal pins or ports.

DESCRIPTION

The **get_related_supply_nets** command creates a collection of related supply nets of the specified collection of pins or ports. Power supply net is returned by default. If the -ground option is specified, ground supply net is returned.

Until commit_upf command is executed and power intent is finalized, power intent of subblocks are not propagated and will not be included in the result.

EXAMPLES

The following example gets the related supply nets of all pins of a level-shifter cell.

```
prompt> get_related_supply_nets [get_pins Multiplier/mult_on_UPF_LS/*]
{VDDL}
```

SEE ALSO

create_supply_net(2)
connect_supply_net(2)
set_related_supply_net(2)

get_repeater_group_info

Get detailed information for input repeater group.

SYNTAX

status **get_repeater_group_info**
-group_id *group_id*
-type cutline | driver_group_id

Data Types

group_id integer

ARGUMENTS

-group_id *group_id*

Specifies the group id set by set_repeater_group.

-type

Specifies the type to query, cutline or driver_group_id.

DESCRIPTION

This command gets detailed information for input repeater group.

EXAMPLES

The following examples show the usages of command.

prompt> **get_repeater_group_info -group_id 4 -type driver_group_id**
prompt> **get_repeater_group_info -group_id 4 -type cutline**

SEE ALSO

set_repeater_group(2)
report_repeater_groups(2)

get_routes_between_objects

Returns net shapes that connect two ports or pins on the same net.

SYNTAX

status **get_routes_between_objects**
[-quiet]
[*object_list*]

Data Types

object_list collection or selection

ARGUMENTS

object_list

Specifies a collection or selection set that contains objects to get routes between. If this option is not specified, global selection set is used.

-quiet

Suppresses all output messages.

DESCRIPTION

This command returns a collection of net shapes that connect two ports or pins on the same net.

If the selection contains two objects of port or leaf pin on the same net, and the two objects are physically connected by routes of the net, this command returns a collection of net shapes that connect these two objects. Otherwise the command returns an empty collection.

EXAMPLES

The following example selects routes that connect the two leaf pins A0_12__reg/Q and RES_I109/ADD_6/A1.

prompt> **change_selection [get_pins {A0_12__reg/Q RES_I109/ADD_6/A1}]**
prompt> **change_selection [get_routes_between_objects]**

The alternative syntax uses collection to specify objects.

prompt> **change_selection [get_routes_between_objects [get_pins {A0_12__reg/Q RES_I109/ADD_6/A1}]]**

SEE ALSO

change_selection(2)
get_selection(2)
gui_select_connections_of_selected(2)

get_routing_blockages

Creates a collection of routing blockages from the current design.

SYNTAX

collection **get_routing_blockages**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[-include_lib_cell]

[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count int
minimum_count int
of_objects list
patterns string
region list
point list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with *expression*. For any routing blockages that match *patterns*, the expression is evaluated based on the routing blockage's attributes. If the expression evaluates to true, the routing_blockage is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for routing blockages level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-include_lib_cell

Search for routing blockages in the **lib_cell**. By default, **lib_cell** routing_blockages are excluded from search.

This option must be used with **-hierarchical** option.

-of_objects *of_objects*

Creates a collection containing the routing blockages of the specified edit groups or cells. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together.

patterns

Matches the routing blockage names in the current design against the specified patterns. You can specify the patterns by using the following formats:

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses *

(asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all routing_blockages at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all routing_blockages that are completely inside the specified region and doesn't include routing_blockages which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all routing_blockages that are completely inside the specified region and the routing_blockages which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all routing_blockages which are abut/touching from inside or outside to the specified region and the routing_blockages whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command returns a collection of routing blockages of the current design that meet the selection criteria. You can also use this command to query routing blockages by filtering their attributes.

EXAMPLES

The following example returns the routing blockages whose area is larger than 1000.

```
prompt> get_routing_blockages * -filter "area > 1000"
{"RB_4683"}
```

The following example returns all routing blockages.

```
prompt> get_routing_blockages "RB_*"
{"RB_4683 RB_5389"}
```

The following example shows how to use **get_attribute** to query the value of the routing blockage attributes.

```
prompt> get_attribute [get_routing_blockages  RB_4683] layer
via3Blockage

prompt> get_attribute [get_routing_blockages RB_4683] bbox
{0.000 0.000{ {100.000 100.000}

prompt> get_attribute [get_routing_blockages RB_4683] area
10000.000000
```

To get a complete report of the attribute values of all routing blockages, use the **report_attributes** command, as shown in the following example.

```
prompt> report_attributes -application [get_routing_blockages RB_4683]
Design  Object   Type   Attribute Name   Value
------------------------------------------------------------------------
CORE    RB_4683  float    area         10000.000000
CORE    RB_4683  string   bbox         {0.000 0.000} {100.000 100.000}
CORE    RB_4683  string   full_name    CORE/RB_4683
CORE    RB_4683  string   layer        via3Blockage
CORE    RB_4683  string   name         RB_4683
CORE    RB_4683  string   object_class  route_guide
```

SEE ALSO

collections(2)
create_routing_blockage(2)
get_attribute(2)
remove_routing_blockages(2)
report_attributes(2)

get_routing_corridor_shapes

Creates a collection of routing corridor shapes from the current design.

SYNTAX

collection **get_routing_corridor_shapes**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns list
objects list
region list
point list

ARGUMENTS

-design *design*

Specifies the top design in which to find objects. If this option is not specified, objects are found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the routing corridor shape attributes. You can determine the routing corridor shape attributes by using the **list_attributes** command. If the expression

evaluates to **true**, the routing corridor shape is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Treats the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

Checks whether each pattern in *patterns* matches at least one object. If one or more patterns does not match, the command raises a Tcl error and command processing will stop.

-hierarchical

Searches for routing corridor shapes level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-of_objects *objects*

Creates a collection containing the routing corridor shapes of the specified routing corridors. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

patterns

Matches the routing corridor shape names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

IP "-at *point*" Creates a collection that contains all routing_corridor_shapes at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all routing_corridor_shapes that are completely inside the specified region and doesn't include routing_corridor_shapes which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all routing_corridor_shapes that are completely inside the specified region and the routing_corridor_shapes which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all routing_corridor_shapes which are abut/touching from inside or outside to the specified region and the routing_corridor_shapes whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of routing corridor shapes in the current design that match the specified criteria.

The command returns a collection if any routing corridor shapes match the specified criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **shell.common.collection_result_display_limit** application option.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples return the routing corridor shapes that match the specified patterns.

```
prompt> get_routing_corridor_shapes CORRIDOR_SHAPE_528896
{ CORRIDOR_SHAPE_528896 }

prompt> get_routing_corridors "CORRIDOR_SHAPE_*"
{ CORRIDOR_SHAPE_6 CORRIDOR_SHAPE_7 CORRIDOR_SHAPE_8
  CORRIDOR_SHAPE_9 }
```

The following example returns the routing corridor shapes that are associated with the corridor_a routing corridor.

```
prompt> get_routing_corridor_shapes \
   -of_objects [get_routing_corridor corridor_a]
{ CORRIDOR_SHAPE_528896 CORRIDOR_SHAPE_528897 CORRIDOR_SHAPE_528898
  CORRIDOR_SHAPE_528899 CORRIDOR_SHAPE_528900 CORRIDOR_SHAPE_528901
  CORRIDOR_SHAPE_528902 CORRIDOR_SHAPE_528903 CORRIDOR_SHAPE_528904
  CORRIDOR_SHAPE_528905 }
```

SEE ALSO

add_to_routing_corridor(2)
collections(2)
create_routing_corridor(2)
create_routing_corridor_shape(2)
filter_collection(2)
get_routing_corridors(2)
query_objects(2)
regexp(2)
remove_from_routing_corridor(2)
remove_routing_corridor_shapes(2)
remove_routing_corridors(2)
report_routing_corridors(2)
shell.common.collection_result_display_limit(3)

get_routing_corridors

Creates a collection of routing corridors from the current design.

SYNTAX

collection **get_routing_corridors**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns list
objects list
region list
point list

ARGUMENTS

-design *design*

Specifies the top design in which to find objects. If this option is not specified, objects are found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the routing corridor attributes. You can determine the routing corridor attributes by using the **list_attributes** command. If the expression evaluates to **true**, the

routing corridor is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Treats the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

Checks whether each pattern in *patterns* matches at least one object. If one or more patterns does not match, the command raises a Tcl error and command processing will stop.

-hierarchical

Searches for routing corridor shapes level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-of_objects *objects*

Creates a collection containing the routing corridors of the specified routing corridor shapes, nets and supernets. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

patterns

Matches the routing corridor names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all routing_corridors at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all routing_corridor_shapes which are abut/touching from inside or outside to the specified region and the routing_corridor_shapes whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all routing_corridors that are completely inside the specified region and the routing_corridors which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all routing_corridors which are abut/touching from inside or outside to the specified region and the routing_corridors whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses *

(asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of routing corridors in the current design that match the specified criteria.

The command returns a collection if any routing corridors match the specified criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

When issued from the command prompt, the command behaves as though you have called the **query_objects** command to report the objects in the collection. By default, it displays a maximum of 100 objects. You can change this maximum by using the **shell.common.collection_result_display_limit** application option.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples return the routing corridors that match the specified patterns.

```
prompt> get_routing_corridors corridor_a
{ corridor_a }

prompt> get_routing_corridors "*"
{ corridor_a corridor_b }
```

The following example returns the routing corridors that are associated with the nets named N1, N2, and N3.

```
prompt> get_routing_corridors -of_objects [get_nets "N1 N2 N3"]
{ corridor_a }
```

SEE ALSO

create_routing_corridor(2)
remove_routing_corridors(2)
report_routing_corridors(2)
add_to_routing_corridor(2)
remove_from_routing_corridor(2)
get_routing_corridor_shapes(2)

create_routing_corridor_shape(2)
remove_routing_corridor_shapes(2)
collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_routing_guides

Creates a collection of routing guides from the current design.

SYNTAX

collection **get_routing_guides**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design string
expression string
exact_count integer
minimum_count integer
patterns list
point list
region list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the routing guide attributes. You can determine the routing guide attributes by using the **list_attributes** command. If the expression evaluates to **true**, the routing guide is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. The option can be used when searching for objects that contain * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for routing guides level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**.

patterns

Matches routing guide names against patterns. Patterns can include the wildcard characters "*" and "?".

The *patterns*, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all routing_guides at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*,**-within**,**-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all routing_guides that are completely inside the specified region and doesn't include routing_guides which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*,**-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all routing_guides that are completely inside the specified region and the routing_guides which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*,**-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all routing_guides which are abut/touching from inside or outside to the specified region and the routing_guides whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*,**-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of routing guides that meet the selection criteria. It returns a collection of routing guides if one or more routing guides meet the selection criteria. If no routing guides match the selection criteria, it returns an empty string.

You can use the **get_routing_guides** command as an argument to another command or assign its result to a variable. Refer to the example below for more information.

See the **collections** man page for information about working with collections.

EXAMPLES

The following example returns all routing guides:

```
prompt> get_routing_guides *
{RG_0 RG_1 RG_2}
```

The following example uses the **-hierarchical** option to return routing guides within child physical blocks.

```
prompt> get_routing_guides -hierarchical
{RG_0 RG_1 RG_2 mid/RG_0 mid/RG_1 mid/bot/RG_0}
```

The following example returns routing guides with an area greater than 1000.

```
prompt>get_routing_guides * -filter "area > 1000"
{RG_2}
```

SEE ALSO

collections(2)
create_routing_guide(2)
remove_routing_guides(2)

get_routing_rules

Returns a collection of routing rules from the current design.

SYNTAX

collection **get_routing_rules**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *of_objects*]

Data Types

design collection
expression string
exact_count int
minimum_count int
of_objects list
patterns string

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with *expression*. For any routing rule that match the patterns option, the expression is evaluated based on the routing rule's attributes. If the expression evaluates to *true*, the routing rule is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are

mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *of_objects*

Creates a collection containing the routing rules of the specified nets. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches routing rule names against the *patterns* argument. The *patterns* argument can include the wildcard character "*".

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

DESCRIPTION

The **get_routing_rules** command creates a collection of routing rules from the current design that match certain criteria. The command returns a collection of routing rules if any routing rule matches the criteria. If no objects match the criteria, the empty string is returned.

You can use the **get_routing_rules** command at the command prompt, or you can nest it as an argument to another command, such as **report_attributes**. In addition, you can assign the **get_routing_rules** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a collection that contains all the routing rules in the current design.

```
prompt> get_routing_rules
{ndr1 test3}
```

The following example returns a collection of all the routing rules corresponding to a net.

```
prompt> get_routing_rules -of_objects Reset
{ndr3}
```

SEE ALSO

create_routing_rule(2)
collections(2)
query_objects(2)
shell.common.collection_result_display_limit(3)

get_rp_blockages

Creates a collection by selecting relative placement blockages from the current design.

SYNTAX

collection **get_rp_blockages**
[*patterns* | -of_objects *of_objects*]
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]

Data Types

design collection
expression string
exact_count int
minimum_count int
of_objects list or collection
patterns list

ARGUMENTS

patterns

Matches the relative placement blockage names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

-of_objects *of_objects*

Creates a collection containing the relative placement blockages of the specified relative placement groups. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the relative placement blockage attributes. You can determine the relative placement blockage attributes by using the **list_attributes** command. If the expression evaluates to **true**, the relative placement blockage is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for relative placement blockages level-by-level relative to the current design. This option is useful only with *patterns* and not with **-of_objects**.

DESCRIPTION

This command creates a collection of relative placement blockages by selecting relative placement blockages that meet the selection criteria. It returns a collection handle if one or more relative placement blockages meet the selection criteria. If no relative placement blockages match the selection criteria, it returns an empty string.

You can use the **get_rp_blockages** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of relative placement blockages by name pattern matching.

```
prompt> get_rp_blockages rp_blockage*
{rp_blockage1 rp_blockage2 rp_blockage3}
```

The following example creates a collection of relative placement blockages with attribute filtering.

```
prompt> get_rp_blockages -filter {height == 2}
{rp_blockage1}
```

The following example creates a collection of all relative placement blockages.

```
prompt> get_rp_blockages *
{rp_blockage1 rp_blockage2 rp_blockage3 rp_blkg4}
```

The following example creates a collection of parent relative placement blockages of relative placement group rp_group1.

```
prompt> get_rp_blockages -of_objects rp_group1}
{rp_blockage1}
```

SEE ALSO

create_rp_group(2)
add_to_rp_group(2)
remove_from_rp_group(2)
remove_rp_groups(2)
collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_rp_group_objects

Creates a collection by selecting cells or relative placement groups or blockages of given relative placement groups.

SYNTAX

collection **get_rp_group_objects**
rp_group_list
[-cell] | [-rp_group] | [-blockage]
[-hierarchical]
[-row row_number]
[-column column_number]

Data Types

rp_group_list list or collection
row_number integer
column_number integer

ARGUMENTS

rp_group_list

Specifies the list of names or collection of a relative placement groups, from which objects needs to be collected.

-cell

Specifies that cells of given relative placement groups must be collected.

-rp_group

Specifies that relative placement groups of given relative placement groups must be collected.

-blockage

Specifies that relative placement blockages of given relative placement groups must be collected.

-hierarchical

Specifies that objects from all sub-groups of relative placement groups in *rp_group_list* are to be collected. By default, objects in sub-groups are not collected.

-row

Specifies the row from which objects must be collected. By default, objects are collected from all rows of the relative placement group.

-column

Specifies the column from which objects must be collected. By default, objects are collected from all columns of the relative placement group.

DESCRIPTION

The **get_rp_group_objects** command returns a collection of cells, relative placement groups and/or blockages. By default, all objects of relative placement group are collected. If user provides *-cell* or *-rp_group* or *-blockage* then cells or relative placement groups or relative placement blockages will be collected respectively.

EXAMPLES

The following example uses **get_rp_group_objects** to create collection cells.

```
prompt> get_rp_group_objects grp_ripple -cell
{U0 U1 U2 U3}
```

The following example uses **get_rp_group_objects** to create collection relative placement groups.

```
prompt> get_rp_group_objects grp_ripple -rp_group
{grp_ripple_hier1 grp_ripple_hier2}
```

The following example uses **get_rp_group_objects** to create collection relative placement blockages.

```
prompt> get_rp_group_objects grp_ripple -blockage
{grp_ripple_blkg}
```

The following example uses **get_rp_group_objects** to get object at given row and column location.

```
prompt> get_rp_group_objects grp_ripple -row 1 -column 1
{grp_ripple_blkg}
```

The following example uses **get_rp_group_objects** to get objects of a row.

```
prompt> get_rp_group_objects grp_ripple -row 1
{grp_ripple_blkg grp_ripple_hier2 U2 U3}
```

The following example uses **get_rp_group_objects** to get objects of a column.

```
prompt> get_rp_group_objects grp_ripple -column 1
{grp_ripple_blkg grp_ripple_hier1 U0 U1}
```

The following example uses **get_rp_group_objects** to get all objects of relative placement group.

```
prompt> get_rp_group_objects grp_ripple
{grp_ripple_blkg U0 U1 U2 U3 grp_ripple_hier1 grp_ripple_hier2}
```

The following example uses **get_rp_group_objects** to get all objects of relative placement group and its hierarchical relative placement groups.

```
prompt> get_rp_group_objects grp_ripple -hierarchical
{grp_ripple_blkg U0 U1 U2 U3 grp_ripple_hier1 grp_ripple_hier2 U5 U6 U7}
```

SEE ALSO

create_rp_group(2)
get_rp_groups(2)
add_to_rp_group(2)
write_rp_groups(2)

get_rp_groups

Creates a collection by selecting relative placement groups from the current design.

SYNTAX

collection **get_rp_groups**
[*patterns* | -of_objects *of_objects*]
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-top]
[-hierarchical]

Data Types

design collection
expression string
exact_count int
minimum_count int
of_objects list or collection
patterns list

ARGUMENTS

patterns

Matches the relative placement group names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

-of_objects *of_objects*

Creates a collection containing the parent relative placement group of the specified cells, relative placement groups or relative placement blockages. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the relative placement group attributes. You can determine the relative placement group attributes by using the **list_attributes** command. If the expression evaluates to **true**, the relative placement group is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-top

If specified with *patterns* then it specifies that only top level relative placement groups must be collected. The option when specified with *of_objects* collects top most relative placement group in parent hierarchy instead of parent relative placement group.

-hierarchical

Searches for relative placement groups level-by-level relative to the current design. This option is useful only with *patterns* and not with **-of_objects**.

DESCRIPTION

This command creates a collection of relative placement groups by selecting relative placement groups that meet the selection

criteria. It returns a collection handle if one or more relative placement groups meet the selection criteria. If no relative placement groups match the selection criteria, it returns an empty string.

You can use the **get_rp_groups** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of relative placement groups by name pattern matching.

```
prompt> get_rp_groups rp_group*
{rp_group1 rp_group2 rp_group3}
```

The following example creates a collection of relative placement groups with attribute filtering.

```
prompt> get_rp_groups -filter {tiling_type == bit_slice}
{rp_group1}
```

The following example creates a collection of all relative placement groups.

```
prompt> get_rp_groups *
{rp_group1 rp_group2 rp_group3 rp_group4}
```

The following example creates a collection of parent relative placement group of cell inv.

```
prompt> get_rp_groups -of_objects inv
{rp_group1}
```

SEE ALSO

create_rp_group(2)
add_to_rp_group(2)
remove_from_rp_group(2)
remove_rp_groups(2)
collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_safety_register_groups

Creates a collection of safety register groups.

SYNTAX

status **get_safety_register_groups**
[*patterns*]
[-of_objects *cells*]
[-filter *expr*]
[-quiet]
[-regexp]
[-exact]
[-nocase]
[-expect *count*]
[-expect_at_least *count*]
[-expect_each_pattern_matches]

Data Types

cells	collection
count	int

ARGUMENTS

patterns

Finds safety register groups with names matching these patterns

-of_objects *cells*

Gets safety register groups associated with these registers or pins

-filter *expression*

Filters collection with *expression*

-quiet

Does not print any messages

-regexp

Specifies that patterns are full regular expressions

-exact

Specifies that wildcards are considered as plain characters

-nocase

Performs case-insensitive matching

-expect *count*

Expects exactly these many matching objects: count >= 0

-expect_at_least *count*

Expects at least these many matching objects: count >= 1

-expect_each_pattern_matches

Specifies that each pattern must match at least one object

DESCRIPTION

Finds and creates a collection of safety_register_group objects based on the options and filters specified. If no options are specified, it returns all the safety register groups of the current design.

EXAMPLES

The following example creates a collection of safety register group objects based on given pattern with case insensitive matching.

prompt> **get_safety_register_groups -nocase group1***

SEE ALSO

create_safety_register_group(2)
report_safety_register_groups(2)
write_safety_register_script(2)

get_safety_register_rules

Creates a collection of safety register rules.

SYNTAX

status **get_safety_register_rules**
[*patterns*]
[-of_objects *registers*]
[-filter *expr*]
[-quiet]
[-regexp]
[-exact]
[-nocase]
[-expect *count*]
[-expect_at_least *count*]
[-expect_each_pattern_matches]

Data Types

registers collection
count int

ARGUMENTS

patterns

Finds safety register rules with names matching these patterns

-of_objects *registers*

Gets safety register rules associated with these registers or pins

-filter *expression*

Filters collection with *expression*

-quiet

Does not print any messages

-regexp

Specifies that patterns are full regular expressions

-exact

Specifies that wildcards are considered as plain characters

-nocase

Performs case-insensitive matching

-expect *count*

Expects exactly these many matching objects: count >= 0

-expect_at_least *count*

Expects at least these many matching objects: count >= 1

-expect_each_pattern_matches

Specifies that each pattern must match at least one object

DESCRIPTION

Finds and creates a collection of safety register rule objects based on the options and filters specified. If no options are specified, it returns all the safety register rules of the current design.

EXAMPLES

The following example creates a collection of safety_register_rule objects based on given pattern with case insensitive matching.

prompt> **get_safety_register_rules -nocase rule1***

SEE ALSO

create_safety_register_rule(2)
report_safety_register_rules(2)
write_safety_register_script(2)

get_scan_cell_names

Returns a list containing the names of scan cells for the specified option filters.

SYNTAX

list **get_scan_cell_names**
[-chain *chain_name_list*]
[-test_mode *mode_name_list* | all]

Data Types

chain_name_list list
mode_name_list list

ARGUMENTS

-chain *chain_name_list*

When specified, retrieves and returns a list of scan cell names contained in the specified scan chains. By default, the scan cells for all chains are returned.

-test_mode *mode_name_list* | all

Reports only on the specified test mode(s), or on all test modes if **all** is specified. For multiple modes, the lists are concatenated together. By default, the tool reports on the current test mode. If no test modes are defined, the default is to report on the Internal_scan mode.

DESCRIPTION

This command returns a Tcl list of scan cell names (as strings), according to the specified options. The information can include scan paths defined by the **set_scan_path** command, as well as scan paths created by the tool.

By default, the command returns the scan cell names for all chains.

SEE ALSO

current_test_mode(2)
get_scan_chain_names(2)

report_dft(2)
set_dft_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(2)

get_scan_cells_of_chain

Returns a collection containing all scan cells of the specified scan chain.

SYNTAX

collection **get_scan_cells_of_chain**
-chain *chain_name*

Data Types

chain_name string

ARGUMENTS

-chain *chain_name*

Specifies the name of the scan chain.

DESCRIPTION

This command returns a Tcl collection containing all scan cells for the specified scan chain.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example gets all scan cells of the *chain1* scan chain:

prompt> **get_scan_cells_of_chain -chain "chain1"**

SEE ALSO

get_scan_chains(2)

get_scan_chain_collection

Returns a Tcl list containing the names of the SCANDEF stub chains of the design.

SYNTAX

list **get_scan_chain_collection**

ARGUMENTS

This command does not receive any arguments.

DESCRIPTION

This command returns a Tcl list containing the names of the SCANDEF stub chains of the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example gets the scan chain names of the design:

```
prompt> get_scan_chain_collection
{ chain1 2 3 4 }
prompt>
```

SEE ALSO

get_scan_chain_count(2)

get_scan_chain_count

Returns the number of SCANDEF stub chains in the current design.

SYNTAX

int **get_scan_chain_count**

DESCRIPTION

This command returns the number of SCANDEF stub chains in the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example returns the number of scan chains in the design:

```
prompt> get_scan_chain_count
203
```

SEE ALSO

get_cells_of_scan_chain(2)

get_scan_chain_names

Returns a list containing the names of scan chains for the specified option filters.

SYNTAX

list **get_scan_chain_names**
[-test_mode *mode_name_list* | all]

Data Types

mode_name_list list

ARGUMENTS

-test_mode *mode_name_list* | all

Reports only on the specified test mode(s), or on all test modes if **all** is specified. For multiple modes, the lists are concatenated together. By default, the tool reports on the current test mode. If no test modes are defined, the default is to report on the Internal_scan mode.

DESCRIPTION

This command returns a Tcl list of scan chain names (as strings), according to the specified options. The information can include scan paths defined by the **set_scan_path** command, as well as scan paths created by the tool.

SEE ALSO

current_test_mode(2)
get_scan_cell_names(2)
report_dft(2)
set_dft_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_path(2)

get_scan_chain_of_cell

Returns a Tcl list containing the name of the SCANDEF stub chain to which the given cell belongs.

SYNTAX

string **get_scan_chain_of_cell**
-cell *cell_name*

Data Types

cell_name string

ARGUMENTS

-cell *cell_name*

Specifies the name of the cell whose scan chain is looked for.

DESCRIPTION

This command returns a Tcl list containing the name of the SCANDEF stub chain corresponding to the given cell.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example gets the chain name *chain1* corresponding to the cell *blockA/a_cell*.

```
prompt> get_scan_chain_of_cell -cell blockA/a_cell
{ chain1 }
prompt>
```

SEE ALSO

get_scan_chain_count(2)

get_scan_chains

Returns a collection of scan chains from the current design.

SYNTAX

collection **get_scan_chains**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns*
| -of_objects *objects*

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns string
objects list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects are found in the current design.

-filter *expression*

Filters the collection with *expression*. For any scan chain that matches the patterns option, the expression is evaluated based on the scan chain's attributes. If the expression evaluates to *true*, the scan chain is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the

=~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-of_objects *objects*

Creates a collection containing the scan chains of the specified stub_chains, ports, pins or cells. Each object in the list is a name, pattern, or collection.

patterns

Matches scan chains names against the *patterns* argument. The *patterns* argument can include the asterisk wildcard character (*).

The *patterns*, **-of_objects** arguments are mutually exclusive; you can specify only.

DESCRIPTION

The **get_scan_chains** command creates a collection of scan chains from the current design that match certain criteria. The command returns a collection of scan chains if any scan chain matches the criteria. If no objects match the criteria, the empty string is returned.

You can use the **get_scan_chains** command at the command prompt, or you can nest it as an argument to another command, such as **report_scan_chains**. In addition, you can assign the **get_scan_chains** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a collection that contains all the scan chains in the current design.

```
prompt> get_scan_chains
{scan1 scan2}
```

The following example returns a collection of all the scan chains associated with a specified pin.

```
prompt> get_scan_chains -of_objects u1/pin1
{scan1 scan2}
```

SEE ALSO

create_scan_chain(2)

get_scenarios

Creates a collection of scenarios in the current design.

SYNTAX

collection **get_scenarios**
[*patterns* | -of_objects *of_objects*]
[-modes *mode_list*]
[-corners *corner_list*]
[-design *design*]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-quiet]
[-regexp]
[-nocase]
[-filter *expression*]
string *expression*
list *patterns*
list *of_objects*
list *mode_list*
list *corner_list*

ARGUMENTS

patterns

Matches scenario names against patterns. Patterns can include the wildcard characters "*" and "?".

The *patterns* argument; **-of_objects** option; and **-modes** and **-corners** options are mutually exclusive; you can specify only one. If you do not specify any of these, the generated collection contains all scenarios.

-of_objects ***of_objects***

Creates a collection of scenarios that are associated with the specified objects. Each object is a named mode or corner, or mode or corner collection.

The *patterns* argument; **-of_objects** option; and **-modes** and **-corners** options are mutually exclusive; you can specify only one. If you do not specify any of these, the generated collection contains all scenarios.

-modes ***mode_list***

Creates a collection of scenarios that are associated with the specified modes.

If you use both the **-modes** and **-corners** options, the scenarios are associated with one of the specified modes and one of the

specified corners.

You cannot use the **-modes** or **-corners** options with the **-of_objects** option or *patterns* argument. If you do not specify any of these, the generated collection contains all scenarios.

-corners *corner_list*

Creates a collection of scenarios that are associated with the specified corners.

If you use both the **-modes** and **-corners** options, the scenarios are associated with one of the specified modes and one of the specified corners.

You cannot use the **-modes** or **-corners** options with the **-of_objects** option or *patterns* argument. If you do not specify any of these, the generated collection contains all scenarios.

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-filter *expression*

Filters the collection with *expression*. For any scenarios that match *patterns*, the expression is evaluated based on the scenario's attributes. If the expression evaluates to true, the scenario is included in the result.

DESCRIPTION

The **get_scenarios** command creates a collection of scenarios in the current design that match the *patterns* argument, the **-of_objects** option, or the **-modes** and **-corners** options, and that pass the filter (if specified). If no objects match the criteria, the empty string is returned.

With the correct combination of options, this command can return all the scenarios of a given mode or corner, or the single scenario (if any) of a given mode and corner.

EXAMPLES

The following example returns all scenarions.

```
prompt> get_scenarios
{func::corner_S1_max func::corner_S1_min func::corner_S5_max
func::corner_S5_min func::corner_S7_max func::corner_S7_min}
```

The following example returns all scenarios whose name matches the pattern *min.

```
prompt> get_scenarios *min
{func::corner_S1_min func::corner_S5_min func::corner_S7_min}
```

The following example returns all active setup scenarios.

```
prompt> get_scenarios -filter setup&&active
{func::corner_S1_max func::corner_S5_max func::corner_S7_max}
```

The following examples return all scenarios that use a corner that matches the pattern *min.

```
prompt> get_scenarios -corners *min
{func::corner_S1_min func::corner_S5_min func::corner_S7_min}

prompt> get_scenarios -of_objects [get_corners *min]
{func::corner_S1_min func::corner_S5_min func::corner_S7_min}
```

The following example returns the scenarios that use a corner named *min or a mode named func (or both).

```
prompt> get_scenarios -of_objects [add_to_collection \
  [get_corners *min] [get_modes func]]
{func::corner_S1_min func::corner_S5_min func::corner_S7_min
func::corner_S1_max func::corner_S5_max func::corner_S7_max}
```

The following example returns the scenarios that use the func mode and the corner_S7_max corner.

```
prompt> get_scenarios -corners corner_S7_max -modes func
{func::corner_S7_max}
```

SEE ALSO

get_modes(2)
get_corners(2)
collections(2)
all_scenarios(2)
report_scenarios(2)

create_scenario(2)
current_scenario(2)

get_selection

Returns a collection containing the current selection in the GUI.

SYNTAX

collection **get_selection**
[-slct_targets *target_selection_bus*
[-slct_targets_operation *operation*]]
-create_slct_buses
[-name *selection_bus*]
[-type *object_type*]
[-design *design*]
[-more_than *more*]
[-fewer_than *fewer*]
[-count]
[-num *num*]
[-type_list]

Data Types

target_selection_bus string
operation string
selection_bus string
object_type string
design string
more integer
fewer integer
num integer

ARGUMENTS

-slct_targets *target_selection_bus*

Specifies the name of a selection bus in which to store the result. When you use this option, the *target_selection_bus* value is also returned as result of the command. By default, the result is returned in a collection.

-slct_targets_operation *operation*

Specifies which operation should be used when storing the result in the selection bus specified by the *target_selection_bus* argument. The valid values for the *operation* argument are **clear**, **add**, and **remove**. You can use the **-slct_targets_operation** option only if you also use the **-slct_targets** option.

-create_slct_buses

Specifies to create a new selection bus in which to store the result. Using this option is equivalent to using the

create_selection_bus command to create a selection bus and then providing the created selection bus to the **-slct_targets** option.

-name *selection_bus*

Specifies the selection bus from which the selection is taken.

-type *object_type*

Specifies the type of selected object with which to filter the selection. The valid values for *object_type* and their descriptions are as follows:

design -design object
port -port object
net -net object
cell -cell object
pin -pin object
path -timing path object

-design *design*

Returns only objects in the filter specified by the value of *design*.

-more_than *more*

Returns true if the selection bus contains more than the number of objects specified by *more*.

-fewer_than *fewer*

Returns true if the selection bus contains more than the number of objects specified by *fewer*.

-count

Returns the number of elements contained in the selection bus.

-num *num*

Returns only the number of objects specified by *num* or fewer.

-type_list

Returns a Tcl list of the types of elements contained in the selection bus.

DESCRIPTION

This command returns the current selection in the tool as a collection. It returns a collection handle (identifier) if any objects are selected. If no objects are selected, the command returns an empty string. If you do not use the **-type** option, the command returnS a heterogeneous collection of all objects currently selected. If you use **-type**, the collection is filtered to a homogeneous one containing only objects of the type you specify.

You can use the **get_selection** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**), or assign the **get_selection** result to a variable.

When issued from the command prompt, **get_selection** behaves as if you had called **query_objects** to display the objects in the collection. By default, the command displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

The "implicit query" property of **get_selection** provides a fast, simple way to display cells in a collection. However, if you want the

flexibility provided by the options of the **query_objects** command (for example, if you want to display the object class), use **get_selection** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns the collection of the selected cell objects.

```
prompt> get_selection -type cell
{"I_PRGRM_CNT_TOP", "I_DATA_PATH", "I_REG_FILE", "I_STACK_TOP"}
```

The following example assigns the collection to a variable named **collect_a** that is passed to the **query_objects** command.

```
prompt> set collect_a [get_selection -type cell]
{"I_PRGRM_CNT_TOP", "I_DATA_PATH", "I_REG_FILE", "I_STACK_TOP"}
prompt> query_objects $collect_a
{"I_PRGRM_CNT_TOP", "I_DATA_PATH", "I_REG_FILE", "I_STACK_TOP"}
```

SEE ALSO

change_selection(2)
collections(2)
filter_collection(2)
query_objects(2)

get_shape_patterns

Creates a collection by selecting shape_patterns from the block.

SYNTAX

collection **get_shape_patterns**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects
| -touching { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects
| -intersect { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects]

Data Types

design collection
expression string
exact_count integer
minimum_count integer
patterns string
objects list
point list
llx float
lly float
urx float
ury float
x float
y float
geometric_objects collection

ARGUMENTS

-design *design*

Specifies the block in which to find shape_patterns.

By default, the tool finds objects in the current block.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on shape_pattern attributes. You can determine the shape_pattern attributes by using the **list_attributes -class shape_pattern** command. If the expression evaluates to **true**, the shape_pattern is included in the collection.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The **-exact** and **-regexp** options are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Specifies that each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for shape_patterns level-by-level relative to the current instance. You can use this option with the *patterns*, **-within**, **-touching**, **-intersect**, and **-at** arguments, but not with the **-of_objects** option.

-of_objects *of_objects*

Creates a collection that contains the shape_patterns connected to the specified net, layer, edit group, or group. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. However, you can specify the **-within** and **-intersect** options together. If you do not specify any of these arguments, the command

uses * (asterisk) as the *pattern*.

patterns

Matches the shape_pattern names against the patterns in the current block. You can specify the patterns by using the following formats:

- *** (asterisk) specifies all shape_patterns
- *PATH_PATTERN_** specifies all shape_patterns with shape_pattern_type path_pattern
- *RECT_PATTERN_** specifies all shape_patterns with shape_pattern_type rect_pattern

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. However, you can specify the **-within** and **-intersect** options together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at ***point***

Creates a collection that contains all shape_patterns whose region completely contains the input point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. However, you can specify the **-within** and **-intersect** options together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, the query is limited to the current physical hierarchy. To query objects across physical hierarchies, use the **-hierarchical** option.

-within ***region***

Creates a collection that contains all shape_patterns completely contained inside the specified region. All objects which are abut/touching from inside and completely contained also get selected.

When the *design.enable_icc_region_query* application option is set to *true*, only the objects which are completely contained but not abut/touching from inside get selected.

The region can be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates (i.e., {*llx lly*} {*urx ury*}). A polygon is specified by its points (i.e., {*x y*} {*x y*} {*x y*} {*x y*}...).

Polygons can also be specified as the combined area of a heterogeneous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

Region specified can be a disjointed rectangle or polygon.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. However, you can specify the **-within** and **-intersect** options together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, the query is limited to the current physical hierarchy. To query objects across physical hierarchies, use the **-hierarchical** option.

-touching ***region***

Creates a collection that contains all shape_patterns completely contained by the specified region. All objects which are touching from inside and completely contained also get selected.

The region can be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates (i.e., {*llx lly*} {*urx ury*}). A polygon is specified by its points (i.e., {*x y*} {*x y*} {*x y*} {*x y*}...).

Polygons can also be specified as the combined area of a heterogeneous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

Region specified can be a disjointed rectangle or polygon.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. However, you can specify the **-within** and **-intersect** options together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, the query is limited to the current physical hierarchy. To query objects across physical hierarchies, use the **-hierarchical** option.

-intersect *region*

Creates a collection that contains all shape_patterns whose region intersects the specified region. A certain portion of the region should be completely inside and a certain portion outside of the input window to pass this test.

When the *design.enable_icc_region_query* application option is set to *true*, even the objects which are abut/touching from either inside or outside also get selected.

The region can be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates (i.e., {*llx lly*} {*urx ury*}). A polygon is specified by its points (i.e., {*x y*} {*x y*} {*x y*} {*x y*}...).

Polygons can also be specified as the combined area of a heterogeneous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

Region specified can be a disjointed rectangle or polygon.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. However, you can specify the **-within** and **-intersect** options together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, the query is limited to the current physical hierarchy. To query objects across physical hierarchies, use the **-hierarchical** option.

DESCRIPTION

This command creates a collection of shape_patterns by selecting shape_patterns from the nets, layer, edit group, or group that meet the selection criteria. It returns a collection if one or more shape_patterns meet the selection criteria. If no shape_patterns match the selection criteria, it returns an empty string.

You can use the **get_shape_patterns** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** man page for information about working with collections.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets all shape_patterns connected to the specified net.

```
prompt> get_shape_patterns * -of_objects [get_nets VDD]
{PATH_PATTERN_31_0 PATH_PATTERN_31_1}
```

The following example creates a collection of shape_patterns on the M2 layer.

```
prompt> get_shape_patterns -of_objects [get_layers M2]
{RECT_PATTERN_32_0 RECT_PATTERN_32_1 PATH_PATTERN_32_99 ...}
```

SEE ALSO

collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_shapes

Creates a collection by selecting shapes from the block.

SYNTAX

collection **get_shapes**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[-include_fill]
[-include_lib_cell]
[-include_shield]
[-shield_only]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count integer
minimum_count integer
patterns string
objects list
point list
region list

ARGUMENTS

-design *design*

Specifies the block in which to find shapes.

By default, the tool finds objects in the current block.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on shape attributes. You can determine the shape attributes by using the **list_attributes -class shape** command. If the expression evaluates to **true**, the shape is included in the collection.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The **-exact** and **-regexp** options are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Specifies that each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for shapes level-by-level relative to the current instance. You can use this option with the *patterns*, **-within**,\ fB-touching, **-intersect**, and **-at** arguments, but not with the **-of_objects** option.

-include_fill

Search for fill shapes. You must specify the **-hierarchical** option to search the **fill cell hierarchy** instance in the current block.

By default, fill shapes are excluded from search.

-include_lib_cell

Search for shapes in library cells. You must specify the **-hierarchical** option to search for shapes in the library cells.

By default, library cell shapes are excluded from search.

-include_shield

Includes shield shapes when the **-of_objects** option specifies nets. This option is valid only when the **-of_objects** option specifies nets.

This option is mutually exclusive with the **-shield_only** option.

-shield_only

Search only for shield shapes when the **-of_objects** option specifies nets. This option is valid only when the **-of_objects** option specifies nets.

This option is mutually exclusive with the **-include_shield** option.

-of_objects *of_objects*

Creates a collection that contains the shapes connected to the specified net, port, pin, terminal, library cell pin, layer, edit group, or fill cell. Each object in the list is a name, pattern, or collection.

When you specify ports, the **get_shapes** command searches for shapes attached to the terminals associated with the specified ports.

When you specify pins, the **get_shapes** command searches for shapes attached to the terminals of the ports associated with the specified pins.

When you specify fill cells, the **get_shapes** command searches for shapes contained in the specified fill cells.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. However, you can specify the **-within** and **-intersect** options together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

patterns

Matches the shape names against the patterns in the current block. You can specify the patterns by using the following formats:

- *** (asterisk) specifies all shapes
- *PATH_** specifies all path shapes
- *RECT_** specifies all rectangle shapes

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. However, you can specify the **-within** and **-intersect** options together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all shapes at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. However, you can specify the **-within** and **-intersect** options together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, the query is limited to the current physical hierarchy. To query objects across physical hierarchies, use the **-hierarchical** option.

-within *region*

Creates a collection that contains all shapes that are completely inside the specified region and does not include shapes that abut the specified region or touch it from the inside. The region boundary can be a rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. However, you can specify the **-within** and **-intersect** options together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, the query is limited to the current physical hierarchy. To query objects across physical hierarchies, use the **-hierarchical** option.

-touching ***region***

Creates a collection that contains all shapes that are completely inside the specified region and the shapes that abut the specified region or touch it from the inside. The region boundary can be a rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. However, you can specify the **-within** and **-intersect** options together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, the query is limited to the current physical hierarchy. To query objects across physical hierarchies, use the **-hierarchical** option.

-intersect ***region***

Creates a collection that contains all shapes that abut the specified region or touch it from the inside or outside. The region boundary can be a rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one. However, you can specify the **-within** and **-intersect** options together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, the query is limited to the current physical hierarchy. To query objects across physical hierarchies, use the **-hierarchical** option.

DESCRIPTION

This command creates a collection of shapes by selecting shapes from the nets, ports, pins, or layers that meet the selection criteria. It returns a collection if one or more shapes meet the selection criteria. If no shapes match the selection criteria, it returns an empty string.

You can use the **get_shapes** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** man page for information about working with collections.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example gets all shapes connected to the specified net.

```
prompt> get_shapes -of_objects [get_nets net1]
{PATH_31_0 PATH_31_1}
```

The following example creates a collection of shapes on the M2 layer.

```
prompt> get_shapes -of_objects [get_layers M2]
{RECT_32_0 RECT_32_1 PATH_32_99 ...}
```

The following example creates a collection of path shapes.

```
prompt> get_shapes -filter {number_of_points > 2}
{PATH_31_2375 PATH_31_2396 PATH_32_2294 PATH_32_2301}
```

SEE ALSO

collections(2)
filter_collection(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_shaping_blockages

Creates a collection of shaping blockages from the current design.

SYNTAX

collection **get_shaping_blockages**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns* | -of_objects *objects*]

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns string
objects list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects are found in the current design.

-filter *expression*

Filters the collection with *expression*. For any shaping blockages that match *patterns*, the expression is evaluated based on the shaping blockage's attributes. If the expression evaluates to true, the blockage is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the

=~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for shaping blockages level-by-level relative to the current instance. This option is useful only with *patterns* and not with **-of_objects**.

-of_objects *of_objects*

Creates a collection containing the shaping blockages of the specified cells or blocks. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches the shaping blockage names in the current design against the specified patterns. You can specify the patterns by using the following formats:

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses asterisk (*) as the pattern.

DESCRIPTION

This command returns a collection of shaping blockages of the current design that meet the selection criteria. You can use this command to query the shaping blockages inside the database by area, by point, or by region. You can also use this command to query shaping blockages by filtering their attributes.

EXAMPLES

The following example returns the shaping blockages whose area is larger than 1000.

```
prompt> get_shaping_blockages * -filter "area > 1000"
{"SB_4683"}
```

The following example returns all shaping blockages that begin with "SB_".

```
prompt> get_shaping_blockages "SB_*"
{"SB_4683 SB_5389"}
```

The following three commands show how to use **get_attribute** to query the value of the shaping blockage attributes.

```
prompt> get_attribute [get_shaping_blockages SB_4683] layer
via3Blockage

prompt> get_attribute [get_shaping_blockages SB_4683] bbox
{0.000 0.000{ {100.000 100.000}

prompt> get_attribute [get_shaping_blockages SB_4683] area
10000.000000
```

To get a complete report of the attribute values of all shaping blockages, use the **report_attributes** command, as shown in the following example.

```
prompt> report_attributes -application [get_shaping_blockages SB_4683]
Design  Object    Type   Attribute Name   Value
------------------------------------------------------------------------
CORE    SB_4683   float    area         10000.000000
CORE    SB_4683   string   bbox         {0.000 0.000} {100.000 100.000}
CORE    SB_4683   string   full_name    CORE/SB_4683
CORE    SB_4683   string   name         SB_4683
CORE    SB_4683   string   object_class  route_guide
```

SEE ALSO

collections(2)
create_shaping_blockage(2)
get_attribute(2)
remove_shaping_blockages(2)
report_attributes(2)

get_shaping_channels

Get a collection of channels in the current block.

SYNTAX

collection **get_shaping_channels**
[-design design]
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect exact_count]
[-expect_at_least minimum_count]
[-expect_each_pattern_matches]
[-of_objects of_objects]
[-types type list]
[patterns]

Data Types

design design
expression filter_expression
exact_count int
minimum_count int
of_objects collection
type list list of shaping constraint typesi

ARGUMENTS

-design

The -design argument specifies the top design for finding objects.

-filter

The -filter argument filters the collection based on the criteria given in the filter expression.

-quiet

The -quiet argument suppresses all messages.

-regexp

The -regexp argument causes the command to treat the patterns as full regular expressions.

-nocase

The -nocase argument performs case-insensitive matching.

-exact

The -exact argument treats wildcards as plain characters.

-expect exact_count

The -expect argument tells the command to expect exactly this many matching objects.

-expect_at_least minimum_count

The -expect argument tells the command to expect at least this many matching objects.

-expect_each_pattern_matches

The -expect_each_pattern_matches argument requires that each pattern match at least one object.

-of_objects of_objects

The -of_objects argument specifies a list of objects from which to gather the shaping channel objects. The objects may be of type shaping_constraint, in which case the command gathers the shaping channels associated with the given shaping_constraint objects. The objects may also be of types voltage_area, move_bound, cell, shaping group, or block, in which case the command gathers the shaping channels associated with any shaping constraints defined for the given objects.

-types type list

The -types argument specifies a list of channel types to filter the result. The valid types are object and neighbor.

patterns

The patterns argument gives the patterns to match channel names against.

DESCRIPTION

Gathers a returns a collection of channel objects in the current design that match the set of collection criteria.

EXAMPLES

The following command gets all the channel objects.

prompt> get_shaping_channels

The following command gets all the channel objects of type object.

prompt> get_shaping_channels -type object

SEE ALSO

create_shaping_channel(2)
remove_shaping_channels(2)
channel(3)

get_shaping_constraints

Get a collection of shaping_constraint objects

SYNTAX

collection **get_shaping_constraints**
[-design design]
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect exact_count]
[-expect_at_least minimum_count]
[-expect_each_pattern_matches]
[-of_objects of_objects]
[-types type list]
[patterns]

Data Types

design design
expression filter_expression
exact_count int
minimum_count int
of_objects collection
type list list of shaping constraint types

ARGUMENTS

-design

The -design argument specifies the top design for finding objects.

-filter

The -filter argument filters the collection based on the criteria given in the filter expression.

-quiet

The -quiet argument suppresses all messages.

-regexp

The -regexp argument causes the command to treat the patterns as full regular expressions.

-nocase

The -nocase argument performs case-insensitive matching.

-exact

The -exact argument treats wildcards as plain characters.

-expect exact_count

The -expect argument tells the command to expect exactly this many matching objects.

-expect_at_least minimum_count

The -expect argument tells the command to expect at least this many matching objects.

-expect_each_pattern_matches

The -expect_each_pattern_matches argument requires that each pattern match at least one object.

-of_objects of_objects

The -of_objects argument specifies a list of objects from which to gather the shaping_constraint objects.

-types type list

The -types argument specifies a list of shaping constraint types to filter the result. The valid types are alignment_point, allowable_orientation, array_layout, aspect_ratio, boundary_area, boundary_channels, boundary_status, child_default_channels, child_default_utilization, external_channels, is_rigid_boundary, parent_object, reference_region, target_location, utilization.

patterns

The patterns argument gives the patterns to match shaping_constraint names against.

DESCRIPTION

Gathers and returns a collection of shaping_constraint objects in the current design that match the set of selection criteria.

EXAMPLES

The following command gets all the shaping_constraint objects.

```
prompt> get_shaping_constraints
```

The following command gets all the shaping_constraint objects of types target_location and alignment_point.

```
prompt> get_shaping_constraints -types "target_location alignment_point"
```

SEE ALSO

create_shaping_constraint(2)
remove_shaping_constraints(2)
shaping_constraint(3)

get_site_arrays

Returns a collection of site arrays from the current design.

SYNTAX

collection **get_site_arrays**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns string
objects list
region list
point list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects are found in the current design.

-filter *expression*

Filters the collection with *expression*. For any site array that matches the patterns option, the expression is evaluated based on the site array's attributes. If the expression evaluates to *true*, the site array is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for site arrays level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-of_objects *objects*

Creates a collection containing the site arrays of the specified voltage areas and site rows. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

patterns

Matches site array names against the *patterns* argument. The *patterns* argument can include the asterisk wildcard character (*).

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all site_arrays at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all site_arrays that are completely inside the specified region and doesn't include site_arrays which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all site_arrays that are completely inside the specified region and the site_arrays which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all site_arrays which are abut/touching from inside or outside to the specified region and the site_arrays whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical

hierarchies.

DESCRIPTION

The **get_site_arrays** command creates a collection of site arrays from the current design that match certain criteria. The command returns a collection of site arrays if any site array matches the criteria. If no objects match the criteria, the empty string is returned.

You can use the **get_site_arrays** command at the command prompt, or you can nest it as an argument to another command, such as **report_attributes**. In addition, you can assign the **get_site_arrays** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a collection that contains all the site arrays in the current design.

```
prompt> get_site_arrays
{array_1 array_2 array_3}
```

The following example returns a collection of all the site arrays associated with a specified voltage area.

```
prompt> get_site_arrays -of_objects VA1
{SA1_1 SA1_2}
```

SEE ALSO

create_site_array(2)
collections(2)
query_objects(2)
shell.common.collection_result_display_limit(3)

get_site_defs

Returns a collection of site defs from the tech of current or specified library.

SYNTAX

collection **get_site_defs**
[-library *library*]
[-tech *tech_object*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *of_objects*]

Data Types

library string or collection
tech_object collection
expression string
exact_count int
minimum_count int
of_objects list
patterns string

ARGUMENTS

-library *library*

Site definition will be searched in the technology of this library. Default is current library's tech. This is mutually exclusive with -tech option.

-tech *tech_object*

Site definition will be searched in this technology. Default is current library's tech. This is mutually exclusive with -library option.

-filter *expression*

Filters the collection with *expression*. For any site def that match the patterns option, the expression is evaluated based on the site def's attributes. If the expression evaluates to *true*, the site def is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *of_objects*

Creates a collection containing the site defs of the specified site rows, site arrays and lib cells. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches site def names against the *patterns* argument. The *patterns* argument can include the wildcard character "*".

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

DESCRIPTION

The **get_site_defs** command creates a collection of site defs from the tech of the current or specified library that match certain criteria. The command returns a collection of site defs if any site def matches the criteria. If no objects match the criteria, the empty string is returned.

You can use the **get_site_defs** command at the command prompt, or you can nest it as an argument to another command, such as **report_attributes**. In addition, you can assign the **get_site_defs** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a collection that contains all the site defs in the tech of current library.

```
prompt> get_site_defs
{SD_1 SD_2 SD_3}
```

The following example returns a collection of all the site defs belonging to a site array.

```
prompt> get_site_defs -of_objects SA1
{SD1_1}
```

SEE ALSO

collections(2)
query_objects(2)
shell.common.collection_result_display_limit(3)

get_site_rows

Returns a collection of site rows from the current design.

SYNTAX

collection **get_site_rows**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[-first_row]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns string
objects list
region list
point list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects are found in the current design.

-filter *expression*

Filters the collection with *expression*. For any site row that match the patterns option, the expression is evaluated based on the

site row's attributes. If the expression evaluates to *true*, the site row is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for site rows level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-first_row

This will give first site-row with respect to core area. First site-row will be left-most for vertical site-rows, or the bottom-most for horizontal site-rows with respect to core area bounding box.

-of_objects *of_objects*

Creates a collection containing the site rows of the specified edit groups and site arrays. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

patterns

Matches site row names against the *patterns* argument. The *patterns* argument can include the asterisk wildcard character (*).

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all site_rows at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all site_rows that are completely inside the specified region and doesn't include site_rows which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all site_rows that are completely inside the specified region and the site_rows which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all site_rows which are abut/touching from inside or outside to the specified region and the site_rows whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

The **get_site_rows** command creates a collection of site rows from the current design that match certain criteria. The command returns a collection of site rows if any site row matches the criteria. If no objects match the criteria, the empty string is returned.

You can use the **get_site_rows** command at the command prompt, or you can nest it as an argument to another command, such as **report_attributes**. In addition, you can assign the **get_site_rows** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a collection that contains all the site rows in the current design.

```
prompt> get_site_rows
{ROW_1 ROW_2 ROW_3}
```

The following example returns a collection of all the site rows that belong to site array SA1.

```
prompt> get_site_rows -of_objects SA1
{SA1_1 SA1_2}
```

SEE ALSO

collections(2)
query_objects(2)
shell.common.collection_result_display_limit(3)

get_snap_setting

Returns the current snap setting for the specified option.

SYNTAX

status **get_snap_setting**
-enabled |
-class *obj_class* |
-cursor_edge |
-object_edge |
-edge_radius |
-preferred_track |
-fix_orientation |
-macro_by_color |
-user_grid

Data Types

obj_class string

ARGUMENTS

-enabled

Returns the current setting for **set_snap_setting -enabled**: 1 if snapping is enabled, 0 if snapping is disabled.

-class *obj_class*

Returns the current setting for **set_snap_setting -class class_name -snap object**. Valid class values are std_cell, macro_cell, io_cell, metal_shape, edit_group, placement_constraint, wiring_constraint, or other.

-cursor_edge

Returns the current setting for **set_snap_setting -cursor_edge**: 1 if objects should be snapped to visible object edges, 0 otherwise.

-object_edge

Returns the current setting for **set_snap_setting -object_edge**: 1 if moved or stretched object should be snapped to other object edges, 0 if not.

-edge_radius

Returns the current integer value setting for **set_snap_setting -edge_radius**, which specifies the search radius for snapping to object edges.

-preferred_track

Returns the current setting for **set_snap_setting -preferred_track**: 1 if only preferred tracks should be used if snapping to tracks is on, 0 otherwise.

-fix_orientation

Returns the current setting for **set_snap_setting -fix_orientation**: 1 if cell orientation should be fixed to match site rows or I/O guides, 0 otherwise.

-macro_by_color

Returns the current setting for **set_snap_setting -macro_by_color**: 1 if soft and hard macros should be snapped so that their colored pins are snapped to matching colored tracks, 0 otherwise.

-user_grid

Returns the current setting for **set_snap_setting -user_grid**: the command returns nothing if this option is not set.

DESCRIPTION

The command queries the current snap settings in the GUI. You can only query one option at a time.

EXAMPLES

The following example returns 1 if snapping is enabled, 0 if snapping is disabled.

```
prompt> get_snap_setting -enabled
1
```

SEE ALSO

set_snap_setting(2)

get_stub_chains

Returns a collection of stub chains from the current design.

SYNTAX

collection **get_stub_chains**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns*
| -of_objects *objects*

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns string
objects list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects are found in the current design.

-filter *expression*

Filters the collection with *expression*. For any stub chain that matches the patterns option, the expression is evaluated based on the stub chain's attributes. If the expression evaluates to *true*, the stub chain is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the

=~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-of_objects *objects*

Creates a collection containing the stub chains of the specified scan_chains, ports, pins or cells. Each object in the list is a name, pattern, or collection.

patterns

Matches stub chains names against the *patterns* argument. The *patterns* argument can include the asterisk wildcard character (*).

The *patterns*, **-of_objects** arguments are mutually exclusive; you can specify only.

DESCRIPTION

The **get_stub_chains** command creates a collection of stub chains from the current design that match certain criteria. The command returns a collection of stub chains if any stub chain matches the criteria. If no objects match the criteria, the empty string is returned.

You can use the **get_stub_chains** command at the command prompt, or you can nest it as an argument to another command, such as **report_stub_chains**. In addition, you can assign the **get_stub_chains** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a collection that contains all the stub chains in the current design.

```
prompt> get_stub_chains
{stub1 stub2}
```

The following example returns a collection of all the stub chains associated with a specified pin.

```
prompt> get_stub_chains -of_objects u1/pin1
{stub1 stub2}
```

SEE ALSO

create_stub_chain(2)

get_supernets

Creates a collection of supernets from the netlist. You can assign these supernets to a variable or pass them into another command.

SYNTAX

collection **get_supernets**
[-design *design*]
[-hierarchical]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hsc *separator*]
patterns | -of_objects *objects*

Data Types

design string
expression string
exact_count integer
minimum_count integer
separator character
patterns list
objects list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this option is not specified, objects will be found in the current design.

-hierarchical

Searches for supernets level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX **find** command. For example, if there is a supernet block1/snet1, a hierarchical search finds it using "snet1". This option is useful with *patterns* but not with **-of_objects**.

-filter *expression*

Filters the collection with *expression*. For any supernets that match *patterns* (or *objects*), the expression is evaluated based on the attributes on the supernet. If the expression evaluates to true, the supernet is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Considers the *patterns* argument as true regular expressions rather than simple wildcard patterns. This option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

Specifies that each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hsc *separator*

Specifies a separator character. The default is /. Valid values are /, @, ^, #, ., and |.

-of_objects *objects*

Creates a collection of supernets connected to the specified objects. Each object is either a named pin, a pin collection, a named cell, cell collection, a named net or a net collection, a named bundle or a bundle collection, a named routing corridor or a routing corridor collection. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

patterns

Matches supernet names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option. Patterns can also include collections of type supernet. *patterns* and **-of_objects** are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **get_supernets** command creates a collection of supernets in the current design, relative to the current instance that match certain criteria. The command returns a collection if any supernets match the *patterns* or *objects* and pass the filter (if specified). If no objects matched the criteria, the empty collection is returned.

If any *patterns* (or *objects*) fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_supernets** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_supernets** result to a variable.

When issued from the command prompt, **get_supernets** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the **shell.common.collection_result_display_limit** application option.

The "implicit query" property of **get_supernets** provides a fast, simple way to display supernets in a collection. However, if you want the flexibility provided by the **query_objects** options (for example, if you want to display the object class), use **get_supernets** as an argument to **query_objects**.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries the supernets that begin with 'SUPERNET' in block 'block1'. Although the output looks like a list, it is just a display.

```
prompt> get_supernets block1/SUPERNET
{"block1/NET1QNX", "block1/NET2QNX"}
```

The following example queries the supernets that are anchored to a collection of pins.

```
prompt> current_instance block1
block1
prompt> set pinsel [get_pins {o_reg1/QN o_reg2/QN}]
{"o_reg1/QN", "o_reg2/QN"}
prompt> query_objects [get_supernets -of_objects $pinsel]
{"SUPERNET1QNX", "SUPERNET2QNX"}
```

SEE ALSO

collections(2)
filter_collection(2)
create_supernet(2)
remove_supernets(2)
query_objects(2)
regexp(2)
shell.common.collection_result_display_limit(3)

get_supply_net_probability

Gets switching probability on selected supply nets in the current design.

SYNTAX

string **get_supply_net_probability**
[-scenarios *scenario_list*]
[-modes *mode_list*]
[-corners *corner_list*]
object_list

Data Types

scenario_list	list
mode_list	list
corner_list	list
object_list	list

ARGUMENTS

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Output of the command is generated separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command reports for the current scenario in the design.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering on modes will occur.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If this option is not specified, no filtering on corners will occur.

object_list

Specifies a supply net in the current design for which the value of the static probability are to be reported.

DESCRIPTION

Use this command to report the static probability of supply nets in the current design.

The supply nets that are reported can be specified explicitly as a list of objects.

If a combination of a mode and corner is specified that does not represent a valid scenario, those are ignored.

Multicorner-Multimode Support

EXAMPLES

The following **get_supply_net_probablity** command reports the static probability on supply net VSS.

```
prompt> get_supply_net_probability [get_supply_net VSS]
(mode = default, corner = default, probability = 1, type = default)
```

SEE ALSO

set_supply_net_probability(2)
reset_supply_net_probability(2)
power.scale_leakage_power_at_power_off(3)
power.scale_dynamic_power_at_power_off(3)

get_supply_nets

Creates a collection of supply nets that meet the specified criteria.

SYNTAX

collection **get_supply_nets**
[-hierarchical]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-of_objects *objects*]
[-top_net_of_hierarchical_group]
[-segments]
[[*patterns*]

Data Types

expression string
exact_count int
minimum_count int
objects list
patterns list

ARGUMENTS

-hierarchical

Searches for supply nets in the current scope and all of its descendant scopes. If you specify this option, you can specify only simple names in the *patterns* argument; the name cannot contain hierarchy delimiter characters.

If you do not specify this option, the tool searches for supply nets only in the current scope and you can use hierarchy delimiter characters in the *patterns* argument to specify hierarchical supply net names relative to the current scope.

You cannot specify this option when you use the **-of_objects** option.

-filter *expression*

Filters the collection with *expression*. For any supply nets that match *patterns*, the expression is evaluated based on the supply net's attributes. If the expression evaluates to true, the supply net is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *objects*

Returns a collection of supply nets associated with the specified objects. The objects can be supply ports, pg leaf pins, voltage_areas, or supply sets.

The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either, the tool uses a pattern of "*".

-top_net_of_hierarchical_group

Keeps only the top supply_net of a hierarchical group. When more than one hierarchical supply_net of the same group is specified (equivalent supply_net at various hierarchical levels), only the supply_net closest to the top of the hierarchy is saved with the collection. In the case of multiple supply_nets at the same level, the first supply_net specified is kept. Although this option can be used when there are multiple nets specified, it is best used in combination with **-segments** and with a single supply_net.

-segments

Returns all equivalent supply_nets for given supply_nets. This modifies the initial search which matches *patterns* or *objects*. It is applied after filtering, but before the **-top_net_of_hierarchical_group** is determined. For each net, all equivalent supply_nets will be added to the result collection. Although this option can be used with other options and when there are multiple supply_nets specified, it is best used with a single net. When combined with the **-top_net_of_hierarchical_group** option, you can isolate the highest scope of equivalent supply_net.

patterns

Matches supply net names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, refer to the **wildcards** man page.

The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either, the tool uses "*".

DESCRIPTION

The **get_supply_nets** command creates a collection of supply nets in the current design, that match certain criteria. If no supply nets match the criteria, an empty string is returned.

Until the **commit_upf** command is executed and power intent is finalized, supply nets of subblocks are not propagated and will not be included in the result.

Regular expression matching is the same as in the Tcl regexp command. When using **-regexp**, take care in the way you quote the patterns and filter expression. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_supply_nets** command at the command prompt, or you can nest it as an argument to another command, such as **report_supply_nets**. In addition, you can assign the **get_supply_nets** result to a variable.

When issued from the command prompt, **get_supply_nets** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum by setting the **shell.common.collection_result_display_limit** application option.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example gets all supply nets under the current scope.

```
prompt> get_supply_nets -hierarchical
{"VDD", "SUB_INST/VDD"}
```

SEE ALSO

create_supply_net(2)
report_supply_nets(2)
shell.common.collection_result_display_limit(3)

get_supply_ports

Creates a collection of supply ports that meet the specified criteria.

SYNTAX

collection **get_supply_ports**
[-hierarchical]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-of_objects *objects*]
[-exclude_power_switch]
[[*patterns*]

Data Types

expression string
exact_count int
minimum_count int
objects list
patterns list

ARGUMENTS

-hierarchical

Searches for supply ports in the current scope and all of its descendant scopes. If you specify this option, you can specify only simple names in the *patterns* argument; the name cannot contain hierarchy delimiter characters.

If you do not specify this option, the tool searches for supply ports only in the current scope and you can use hierarchy delimiter characters in the *patterns* argument to specify hierarchical supply port names relative to the current scope.

You cannot specify this option when you use the **-of_objects** option.

-filter *expression*

Filters the collection with *expression*. For any supply ports that match *patterns*, the expression is evaluated based on the supply port's attributes. If the expression evaluates to true, the supply port is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *objects*

Returns a collection of supply ports associated with the specified supply net objects.

The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either, the tool uses a pattern of "*".

-exclude_power_switch

Exclude the supply ports of power switches. By default, supply ports of power switches are included in the result.

patterns

Matches supply port names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, refer to the **wildcards** man page.

The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either, the tool uses "*".

DESCRIPTION

The **get_supply_ports** command creates a collection of supply ports in the current design, that match certain criteria. If no supply ports match the criteria, an empty string is returned.

Until commit_upf command is executed and power intent is finalized, supply ports of subblocks are not propagated and will not be included in the result.

Regular expression matching is the same as in the Tcl regexp command. When using **-regexp**, take care in the way you quote the patterns and filter expression. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_supply_ports** command at the command prompt, or you can nest it as an argument to another command, such as **report_supply_ports**. In addition, you can assign the **get_supply_ports** result to a variable.

When issued from the command prompt, **get_supply_ports** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum by using the **shell.common.collection_result_display_limit** application option.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example gets all supply ports under the current scope.

```
prompt> get_supply_ports -hierarchical
{"VDD", "SUB_INST/VDD"}
```

SEE ALSO

create_supply_port(2)
report_supply_ports(2)
shell.common.collection_result_display_limit(3)

get_supply_sets

Creates a collection of supply sets that meet the specified criteria.

SYNTAX

collection **get_supply_sets**
[-hierarchical]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-of_objects *objects*]
[[*patterns*]

Data Types

expression string
exact_count int
minimum_count int
objects list
patterns list

ARGUMENTS

-hierarchical

Searches for supply sets in the current scope and all of its descendant scopes. If you specify this option, you can specify only simple names in the *patterns* argument; the name cannot contain hierarchy delimiter characters.

If you do not specify this option, the tool searches for supply sets only in the current scope and you can use hierarchy delimiter characters in the *patterns* argument to specify hierarchical supply net names relative to the current scope.

You cannot specify this option when you use the **-of_objects** option.

-filter *expression*

Filters the collection with *expression*. For any supply sets that match *patterns*, the expression is evaluated based on the attributes of the supply set. If the expression evaluates to true, the supply set is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *objects*

Returns a collection of supply sets associated with the specified objects. The objects can be power domains and supply nets.

The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either, the tool uses a pattern of "*".

patterns

Matches supply set names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, refer to the **wildcards** man page.

The *patterns* argument and **-of_objects** option are mutually exclusive. If you do not specify either, the tool uses "*".

DESCRIPTION

The **get_supply_sets** command creates a collection of supply sets in the current design, that match certain criteria. If no supply sets match the criteria, an empty string is returned.

Until commit_upf command is executed and power intent is finalized, supply sets of subblocks are not propagated and will not be included in the result.

Regular expression matching is the same as in the Tcl regexp command. When using **-regexp**, take care in the way you quote the patterns and filter expression. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_supply_sets** command at the command prompt, or you can nest it as an argument to another command, such as **report_supply_sets**. In addition, you can assign the **get_supply_sets** result to a variable.

When issued from the command prompt, **get_supply_sets** behaves as though **query_objects** has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum by using the **shell.common.collection_result_display_limit** application option.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example gets all supply sets under the current scope.

```
prompt> get_supply_sets
{SS1 SS2 SS3 SS4 SS5 new_sset_1}
```

The following example gets all supply sets matching a given pattern.

```
prompt> get_supply_sets SS*
{SS1 SS2 SS3 SS4 SS5}
```

SEE ALSO

create_supply_set(2)
report_supply_sets(2)
shell.common.collection_result_display_limit(3)

get_svf

Get the name of the current SVF directory.

SYNTAX

boolean **get_svf**

DESCRIPTION

This command prints the name of the SVF directory that is currently open for recording SVF guidance for Formality. If no SVF directory has been specified using the **set_svf** command, a message is printed.

EXAMPLES

In this example, we query the name of the SVF directory and find out that setup information is not being recorded yet. Then we specify the directory name and repeat the query.

```
prompt> get_svf
SVF guidance is not enabled.
1
prompt> set_svf cache_ctrl_svf
1
prompt> get_svf
The current SVF directory is /project/chip/cache_ctrl/cache_ctrl_svf.
1
```

SEE ALSO

set_svf(2)
Formality User Guide

get_switching_activity

Gets switching activity information on nets, pins, ports and cells in the current design.

SYNTAX

string **get_switching_activity**
[-state_condition *state_condition*]
[-path_sources *path_sources*]
[-rise]
[-fall]
[-related_clock]
[-scenarios *scenario_list*]
[-modes *mode_list*]
[-corners *corner_list*]
object_list

Data Types

state_condition	string
path_sources	list
scenario_list	list
mode_list	list
corner_list	list
object_list	list

ARGUMENTS

-state_condition *state_condition*

Specifies the state condition for reporting state-dependent activities. State dependent activities can be reported when the object is already annotated with state-dependent activities. The state condition specified with this argument must match to a state condition already annotated for this object. Use -state_condition default to specify the default state condition.

-path_sources *path_sources*

Specifies the path sources when reporting path-dependent activities. This can be used when the object is already annotated with path-dependent activities. The path sources specified with this argument must be the same as those annotated for this object. If -path_sources option is used to specify the path_sources value, then -state_condition option must be specified to specify the value of the path_sources.

-rise

This option is used with the **-state_condition** option. It reports rise transition.

-fall

This option is used with the **-state_condition** option. It reports fall transition.

-related_clock

This option is used to print the clock domain of the objects.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Output of the command is generated separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command reports for the current scenario in the design.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering on modes will occur.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If this option is not specified, no filtering on corners will occur.

object_list

Specifies a net, pin, port or cell in the current design for which the values of static probability, the toggle rate and the type of activity are to be reported.

DESCRIPTION

This command is used to report switching activity information on nets, ports, pins and cells. Switching activity information includes toggle rate, static probability and activity type.

Note that *object_list* argument should always be provided to the command. The activity type reported by the command can only be one of 'annotated', 'simulated', 'propagated', 'sta' or 'default'.

For more statistics on the switching activity annotation on the current design, use the **report_switching_activity** command.

Multicorner-Multimode Support

EXAMPLES

The following **get_switching_activity** command reports the toggle rate and static probability values on a pin for the current scenario in the design.

```
prompt> get_switching_activity {H1/p1}
 (mode = default, corner = default, probability = 0.9, toggle_rate = 0.3, type = annotated)
```

The following **get_switching_activity** command reports the activity information for a list of objects for the default state_condition for the current scenario scn1. If the object_list contains nets then the tool will ignore the state_condition option for those nets.

```
prompt> get_switching_activity {H1 H1/p1 H1/n1} -state_condition default
scn1 {{condition default probability 1}}
```

```
scn1 {{condition default rise_toggle_rate 0 fall_toggle_rate 0}}
(mode = scn1.mode, corner = scn1.corner, probability = 0.5,
toggle_rate = 0.0333333, type = default)
```

SEE ALSO

read_saif(2)
report_power(2)
report_switching_activity(2)
reset_switching_activity(2)
set_switching_activity(2)

get_taps

Retrieves a collection of tap objects.

SYNTAX

collection **get_taps**
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns*]

Data Types

string *expression*
list *patterns*
int *exact_count*
int *minimum_count*

ARGUMENTS

-filter *expression*

Filters the collection with the value specified by expression. For any taps that match the *patterns* argument, the expression is evaluated based on the specified tap attributes. If the expression is evaluated to be true, the tap is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Uses the value of the pattern arguments as real regular expressions rather than simple wildcard patterns. This option also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions, rather than with simple wildcard patterns. Note that the -regexp and -exact options are mutually exclusive.

-nocase

Makes the pattern matching case-insensitive.

-exact

Disables the simple pattern matching. This is used when searching for objects that contain wildcard characters, like asterisks (*) and question marks (?) symbols. The -exact and -regexp options are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

Matches each pattern in *patterns* at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

patterns

Matches tap names against patterns. Patterns can include wildcard characters asterisks (*) and question marks (?) symbols or regular expressions, depending on the use of the -regexp option.

DESCRIPTION

The **get_taps** command retrieves a collection of taps that are currently in the session context of the current design. The command returns a collection of the taps that match the patterns and pass the filter criteria (if specified). If no object matches the criteria, an empty collection is returned.

Use the -regexp option to enable regular expression matching in the same way as using the TCL regexp command. When using the -regexp option, be careful of the way you quote the patterns and filter expression. Use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression matching begins at the beginning of an object name and stops at the end of an object name. You can widen the search simply by adding the period and asterisk symbols (.*) to the beginning or end of the expressions as needed.

You can use the **get_taps** command at the command prompt, or nest it as an argument to another command, like the **query_objects** command. In addition, you can assign the result of the **get_taps** run to a variable.

When issued from the command prompt, the **get_taps** command behaves as though the **query_objects** command has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. To change the setting, use the *collection_result_display_limit* variable.

The "implicit query" property of the **get_taps** command provides a fast and simple way to display cells in a collection. However, if you want the flexibility provided by **query_objects** (for example, if you want to display the object class), use the **get_taps** command as an argument to the **query_objects** command.

EXAMPLES

The following example queries all taps that match a pattern TAP?2, except for the TAP_42.

```
prompt> get_taps -filter full_name!=TAP_42 TAP_?2
{"TAP_12", "TAP_22", "TAP_32", "TAP_52"}
```

The following example queries all taps.

```
prompt> get_taps

{"TAP_1", "TAP_2", "TAP_3", "TAP_4"}
```

SEE ALSO

create_taps(2)
remove_taps(2)
report_taps(2)

get_techs

Creates a collection of technology data objects loaded into memory. You can assign these technology data objects to a variable or pass them into another command.

SYNTAX

collection **get_techs**
[-all]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *objects*]

Data Types

expression string
objects list
patterns list

ARGUMENTS

-all

Returns the technology data objects of both the explicitly opened libraries and their reference libraries.

By default, the command returns only the technology data objects of the explicitly opened libraries.

-filter *expression*

Filters the collection with *expression*.

For any technology data objects that match *patterns* (or *objects*), the expression is evaluated based on the technology data object's attributes. If the expression evaluates to true, the technology data object is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modify the behavior of the =~

and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

The **-regexp** and **-exact** options are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters.

The **-exact** and **-regexp** options are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *objects*

Creates a collection of technology data objects of the specified objects. In this case, each object is either a library or a block collection. In case of block, the technology data object of the containing library will be returned. **-of_objects** and *patterns* are mutually exclusive; you must specify one, but not both.

patterns

Matches technology data object names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

DESCRIPTION

The **get_techs** command creates a collection of technology data objects from those currently loaded into memory that match certain criteria. The command returns a collection if any of the technology data objects match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl **regexp** command. When using **-regexp**, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the **get_techs** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **get_techs** result to a variable.

When issued from the command prompt, **get_techs** behaves as though **query_objects** had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the

shell.common.collection_result_display_limit application option.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example creates a technology data object in the specified library. The following **get_techs** command returns the created technology data object.

```
prompt> create_tech -library r4000 tcbn90g
{r4000}
prompt> get_techs -of_objects [get_libs r4000]
{tcbn90g}
```

The following example replaces the existing technology data object by creating a new technology data object in the current library. The following **get_techs** command returns the created technology data object.

```
prompt> create_tech -force tcbn90g
{tcbn90g}
prompt> get_techs -of_objects [current_lib]
{tcbn90g}
```

SEE ALSO

collections(2)
create_tech(2)
filter_collection(2)
query_objects(2)
regexp(2)
remove_tech(2)
shell.common.collection_result_display_limit(3)

get_terminals

Creates a collection by selecting terminals from the design.

SYNTAX

collection **get_terminals**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[-include_lib_cell]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count int
minimum_count int
of_objects list
patterns list
region list
point list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with the specified expression. For each terminal in the collection, the expression is evaluated based on the

terminal's attributes. If the expression evaluates to true, the terminal is included in the result.

To see the list of terminal attributes that you can use in the expression, use the **list_attributes -application -class terminal** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive. use only one.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-exact

Considers wildcards to be plain characters, and does not interpret their meaning as wildcards.

The **-regexp** and **-exact** options are mutually exclusive; you can use only one.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for terminal level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-include_lib_cell

Search for terminals in the **lib_cell**. By default, **lib_cell** vias are excluded from search.

This option must be used with the **-hierarchical** option.

patterns

Creates a collection of terminals whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page. Pattern matching is case sensitive unless you use the **-nocase** option.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-of_objects *port_list*

Creates a collection of terminals connected to the specified ports.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all terminals at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all terminals that are completely inside the specified region and doesn't include terminals which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all terminals that are completely inside the specified region and the terminals which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all terminals which are abut/touching from inside or outside to the specified region and the

terminals whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of terminals by selecting terminals from the specified design or the current design.

The command returns a collection if any terminals match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **get_attribute**. In addition, you can assign the result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of terminals named reset or with a prefix of clock:

```
prompt> get_terminals [list reset clock*]
{reset clock}
```

The following example creates a collection of terminals with attribute filtering.

```
prompt> get_terminals  -filter {port_name == reset}
{reset}
```

The following example creates a collection of all terminals belonged to the port "reset".

```
prompt> get_terminals -of_objects {reset}
{reset}
```

SEE ALSO

collections(2)
create_terminal(2)

filter_collection(2)
list_attributes(2)
query_objects(2)
remove_terminals(2)
shell.common.collection_result_display_limit(3)
wildcards(3)

get_timing_arcs

Creates a collection of timing arcs for custom reporting and other processing.

SYNTAX

collection **get_timing_arcs**
[-design *design*]
[-to *to_list*]
[-from *from_list*]
[-of_objects *object_list*]
[-filter *expression*]
[-quiet]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]

Data Types

design collection
to_list list
from_list list
object_list list
expression string
exact_count integer
minimum_count integer

ARGUMENTS

-design *design*

Specifies the top-level design for finding objects.

By default, objects are found in the current design.

-to *to_list*

Specifies the to pins or to ports to get the timing arcs for. All backward arcs from the specified pins or ports are considered.

You must specify one of the following options: **-to**, **-from**, or **-of_objects**.

-from *from_list*

Specifies the from pins or from ports to get the timing arcs for. All forward arcs from the specified pins or ports are considered.

You must specify one of the following options: **-to**, **-from**, or **-of_objects**.

-of_objects *object_list*

Specifies the cells or nets for which to get the timing arcs. If you specify a cell, all cell arcs of that cell are considered. If you specify a net, all net arcs of that net are considered.

You must specify one of the following options: **-to**, **-from**, or **-of_objects**.

-filter *expression*

Filters the collection with the specified expression.

For any objects that match, the command evaluates the expression based on the timing arc's attributes. If the expression evaluates to true, the timing arc is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-expect *exact_count*

Specifies the expected number of objects to find. If the command finds a different number of objects, it raises a Tcl error and stops processing.

-expect_at_least *minimum_count*

Specifies the minimum number of objects to find. If the command finds fewer objects, it raises a Tcl error and stops processing.

DESCRIPTION

The **get_timing_arcs** command creates a collection of timing arcs for custom reporting or other operations. Use the **foreach_in_collection** command to iterate among the arcs in the collection. You can use the **get_attribute** command to obtain information about the timing arcs. You can also use a filter expression to filter the arcs that satisfy the specified conditions. To see the attributes supported on timing arcs, use the **list_attributes -application -class timing_arc** command.

Multicorner-Multimode Support

This command works only on the mode associated with the current scenario.

EXAMPLES

The following example reports the to-pin and from-pin for the timing arcs of the ffa cell.

```
prompt> set arcs [get_timing_arcs -of_objects [get_cells ffa]]
prompt> foreach_in_collection arc $arcs {
? set fpin [get_attribute $arc from_pin]
? set tpin [get_attribute $arc to_pin]
? set from_pin_name [get_attribute $fpin full_name]
? set to_pin_name [get_attribute $tpin full_name]
? echo $from_pin_name -> $to_pin_name
? }
ffa/CP -> ffa/D
ffa/CP -> ffa/D
ffa/CP -> ffa/Q
ffa/CP -> ffa/QN
```

ffa/CD -> ffa/Q
ffa/CD -> ffa/QN

SEE ALSO

collections(2)
foreach_in_collection(2)
get_attribute(2)
list_attributes(2)

get_timing_paths

Creates a collection of timing paths for custom reporting and other processing. You can assign these timing paths to a variable or pass them into another command.

SYNTAX

string **get_timing_paths**
  [-from *from_list*
    | -rise_from *rise_from_list*
    | -fall_from *fall_from_list*]
  [-to *to_list*
    | -rise_to *rise_to_list*
    | -fall_to *fall_to_list*]
  [-through *through_list*]
  [-rise_through *rise_through_list*]
  [-fall_through *fall_through_list*]
  [-exclude *exclude_list*
    | -rise_exclude *rise_exclude_list*
    | -fall_exclude *fall_exclude_list*]
  [-delay_type max | min | min_max | max_rise | max_fall | min_rise | min_fall]
  [-path_type full_clock | full_clock_expanded]
  [-report_by design | mode | corner | scenario | group]
  [-sort_by group | slack]
  [-nworst *paths_per_endpoint*]
  [-max_paths *max_path_count*]
  [-slack_lesser_than *lesser_slack_limit*]
  [-pba_mode none | path | exhaustive]
  [-start_end_pair]
  [-modes *mode_list*]
  [-corners *corner_list*]
  [-scenarios *scenario_list*]
  [-groups *group_list*]
  [-exception *exception_type*]
  [-include_hierarchical_pins]

Data Types

from_list list
rise_from_list list
fall_from_list list
to_list list
rise_to_list list
fall_to_list list
through_list list
rise_through_list list
fall_through_list list
exclude_list list

rise_exclude_list list
fall_exclude_list list
paths_per_endpoint int
max_path_count int
lesser_slack_limit float
mode_list list
corner_list list
scenario_list list
group_list list

ARGUMENTS

-from *from_list*

Specifies a list of from pins, ports, cells, or clocks to be processed. Path startpoints are typically the input ports or clock pins of registers. If you specify a clock, all startpoints are considered if they are clocked by the clock.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-to *to_list*

Specifies a list of to pins, ports, cells, or clocks to be processed. Path endpoints are typically the output ports or data pins of registers. If you specify a clock, all endpoints are considered if they are constrained by the clock.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path.

-fall_to *fall_to_list*

Same as the **-to** option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-through *through_list*

Specifies a list of through pins, ports, cells, or nets to be processed. Only paths through the named objects are considered.

You can specify many *through_list* groups by using multiple **-through** options. The objects specified within one **-through** option are assumed to be in OR mode. The group of objects specified with multiple **-through** options is assumed to be in AND mode.

If you specify **-through** only one time, the tool generates only the paths that travel through one or more of the objects in the list.

If you specify multiple **-through** options, the tool generates only the paths that travel through one or more of the objects in each list. the tool uses the exact order in which the **-through** options are listed; so to obtain correct results, you must ensure that this

order is the same as that followed by the actual paths in the circuit.

-rise_through *rise_through_list*

Similar to the **-through** option, except that the path must rise through the objects specified.

-fall_through *fall_through_list*

Similar to the **-through** option, except that the path must fall through the objects specified.

-exclude *exclude_list*

Excludes all data paths from, through, or to specified list of pins, ports, nets, and cell instances. If a cell instance is specified, all pins of the cell are excluded. The **-exclude** option

o Takes precedence over the **-from**, **-through**, and **-to** options.

o Is exclusive with the **-rise_exclude** and **-fall_exclude** options.

o Does not apply to borrowing path from the **-trace_latch_borrow** option or clock path from the **-path full_clock** or **full_clock_expanded** option.

Note that this option is intended for use in conjunction with **-from**, **-through** and **-to** options and can lead to long runtime when used on its own, i.e. without other such topological options.

-rise_exclude *rise_exclude_list*

This option is the same as the **-exclude** option, but applies only to paths rising at the named pins, ports, nets, or cell instances.

-fall_exclude *fall_exclude_list*

This option is the same as the **-exclude** option, but applies only to paths falling at the named pins, ports, nets, or cell instances.

-delay_type max | **min** | **min_max** | **max_rise** | **max_fall** | **min_rise** | **min_fall**

Specifies the type of path delay. Valid values are **max** (the default), **min**, **min_max**, **max_rise**, **max_fall**, **min_rise**, or **min_fall**. The "rise" or "fall" suffix in the delay type refers to a rising or falling transition at the path endpoint.

-path_type full_clock | **full_clock_expanded**

Specifies the structure of the generated timing path objects. The allowed values are **full_clock** and **full_clock_expanded**, which create clock objects for the timing_path's **launch_clock_paths** and **capture_clock_paths** attributes. The **full_clock_expanded** value creates clock path objects that include the path between a primary clock and related generated clocks.

-report_by design | **mode** | **corner** | **scenario** | **group**

Specifies the grouping criteria for generating paths. The default is **design**. It is an error to specify both the **-report_by** and **-sort_by** options.

-sort_by group | **slack**

Specifies the sorting criteria for generating paths. Valid values are: **group** or **slack**. It is an error to give both the **-report_by** and **-sort_by** options. This option is deprecated, use the **-report_by** option instead.

-nworst *paths_per_endpoint*

Retrieves the *n* worst paths to endpoint, where *paths_per_endpoint* is >= 1. The default is 1, meaning that only the worst path to an endpoint is considered. Specifying larger values of *paths_per_endpoint* increases runtime.

-max_paths *max_path_count*

Specifies the maximum number of paths to retrieve per path group, where *max_path_count* is >= 1. The default is 1.

-slack_lesser_than *lesser_slack_limit*

Retrieves only paths with a slack less than **lesser_slack_limit**. With this filtering in effect, it might not be possible to find enough paths to satisfy the **-max_paths** and **-nworst** parameters. If this option is specified, unconstrained paths will not be returned, as unconstrained paths do not have a defined slack.

-pba_mode none | path | exhaustive

Specifies one of the following path-based analysis modes:

- **none** - Disables path-based analysis.
- **path** - Performs path-based analysis on paths after they have been gathered.
- **exhaustive** - Performs an exhaustive path-based analysis to determine the worst path-based analysis path set in the design. You cannot use the **exhaustive** mode together with the **-start_end_pair** or option.

-start_end_pair

Retrieves paths for each pair of startpoint and endpoint based on connectivity. This option can lead to long runtime and can lead to generating a huge number of paths depending on the design. By default, this option searches only for paths that are violating. This default can be changed by having an explicit -slack_lesser_than option. The options that do not work with this option are -nworst, -max_paths, -report_by. Unlike other options of the get_timing_paths command, this option causes the paths returned to no longer be sorted based on slack. Instead, paths are arranged based on the endpoint with those sharing the same endpoint appearing next to one another. The maximum number of paths returned is limited to 2000000.

-modes *mode_list*

Specifies the modes for which to generate paths. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option will be processed. If this option is not given, the command will process scenarios of all modes. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command will process every active scenario of the design. It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to generate paths. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option will be processed. If this option is not given, the command will process every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios for which to generate paths. Each of the specified scenarios is processed. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, every scenario of the design will be processed.

-groups *group_list*

Specifies the list of path group names for generating paths. The default is all path groups. If this option is specified, unconstrained paths will not be returned, as unconstrained paths do not belong to any path group.

-exception *exception_type*

Prints user-specified timing exceptions, namely false paths, multicycle paths, and minimum and maximum delays, that are satisfied per timing path being reported. Only "dominant" exception type is acceptable, which is to report the dominant exception constraint used for the reported path.

Note: The additional analysis required per path with the -exceptions option is not trivial. Therefore, using a report_timing command with the -exceptions option is expected to execute slower than the exact same command without this option. The -exceptions option does not work with -path_type short/end/end_detailed option and forces the -input_pins option.

-include_hierarchical_pins

Includes any hierarchical pins that the generated paths pass through. By default, the paths include only ports and actual leaf cell driver and load pins. If a hierarchical pin is a path start or end point, it will of course always be included.

DESCRIPTION

The **get_timing_paths** command creates a collection of timing path objects for custom reporting or other operations. Use the **foreach_in_collection** command to iterate among the timing paths in the collection. You can use the **get_attribute** and **collection** commands to obtain information about the timing paths. A collection of timing_paths can easily be printed by passing it to the **report_timing** command.

If the **time.report_unconstrained_paths** application option is set to **true**, the command will search for unconstrained paths to endpoints for which constrained paths cannot be found. Searching for unconstrained paths can cause significantly longer runtimes.

The following attributes are among those supported on timing path objects:

arrival
capture_clock_paths
check_value
clock_uncertainty
common_path_pessimism
corner
corner_name
crpr_common_point
design_name
driving_cell_adjustment
endpoint
endpoint_clock
endpoint_clock_arrival_close_edge_type
endpoint_clock_arrival_open_edge_type
endpoint_clock_close_edge_arrival
endpoint_clock_close_edge_type
endpoint_clock_close_edge_value
endpoint_clock_is_inverted
endpoint_clock_is_propagated
endpoint_clock_latency
endpoint_clock_name
endpoint_clock_open_edge_arrival
endpoint_clock_open_edge_type
endpoint_clock_open_edge_value
endpoint_clock_period
endpoint_name
full_name
input_delay
launch_clock_paths
lib_name
mode
mode_name
name
network_latency
object_class
path_group
path_group_name
path_type

points
propagation_type
required
slack
source_latency
startpoint
startpoint_clock
startpoint_clock_arrival_open_edge_type
startpoint_clock_is_inverted
startpoint_clock_is_propagated
startpoint_clock_latency
startpoint_clock_name
startpoint_clock_open_edge_arrival
startpoint_clock_open_edge_type
startpoint_clock_open_edge_value
startpoint_clock_period
startpoint_name

Note that the launch_clock_paths and capture_clock_paths attributes are set only if specified by the **-path_type** option.

The following attributes are among those supported on clock_path objects:

clock
clock_edge_type
clock_edge_value
clock_latency
clock_name
corner
corner_name
design_name
driving_cell_adjustment
endpoint
endpoint_name
full_name
lib_name
mode
mode_name
name
object_class
path_type
points
propagation_type
source_clock
source_clock_edge_type
source_clock_edge_value
source_clock_name
source_latency
startpoint
startpoint_name

One attribute of a timing path or clock path is the *points* collection. A timing point object corresponds to a pin or port along the path. Iterate through these points with **foreach_in_collection** and get attributes on them by using the **get_attribute** command. The following attributes are among those for timing points of a timing path or clock path:

arrival
capacitance
delay

delta_delay
fanout
full_name
name
object
object_class
object_name
rise_fall
transition

For information about creating collections and iterating over the elements of a collection, see the **collections** and **foreach_in_collection** man pages.

For a complete list and descriptions of the attributes on timing_path, clock_path, and timing_port objects, see the **timing_path_attributes**, **clock_path_attributes**, and **timing_point_attributes** man pages.

Multicorner-Multimode Support

By default, this command works on all active scenarios. To specify different scenarios, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following command returns the two worst timing paths that rise from reg/latch/CP with slack less than 20.0.

```
prompt> get_timing_paths -nworst 2 -path_type full_clock -permanent \
  -rise_from reg/latch/CP -slack_lesser_than 20.0 -report_by group
```

If you use the **-pba_mode** option, path recalculation (path-based analysis) is used during the path search, and the worst recalculated paths meeting your criteria are returned. This option requires that a path search is performed to ensure that the paths being returned truly are the worst paths. The additional runtime required for this option depends on the amount of timing improvement resulting from path-based analysis. As the timing improvement from path-based analysis increases, the runtime for the path search increases. Large **-nworst** values can significantly increase the time needed for the search.

SEE ALSO

collections(2)
foreach_in_collection(2)
get_attribute(2)
group_path(2)
report_timing(2)
clock_path_attributes(3)
timing_path_attributes(3)
timing_point_attributes(3)
time.report_unconstrained_paths(3)

get_topological_constraints

Creates a collection of topological pin feedthrough constraints from the current design.

SYNTAX

collection **get_topological_constraints**
[-hierarchical]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *of_objects*]

Data Types

expression string
exact_count integer
minimum_count integer
patterns list
of_objects list

ARGUMENTS

-hierarchical

Returns topological pin constraints for the entire design hierarchy. By default, the command returns constraints for the current level.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the topological constraint attributes. You can determine the topological constraint attributes by using the **list_attributes** command. If the expression evaluates to **true**, the topological constraint is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the

=~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *of_objects*

Creates a collection containing the topological constraints associated with the specified objects. The objects can be constraint owner objects of type net/bundle, or the objects can be the start/end objects. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches the topological constraint names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the *pattern*.

DESCRIPTION

This command creates a collection of topological constraints from the current design that match certain criteria. The command returns a collection handle (identifier) if any topological constraints match the value of the *patterns* option and pass the filter (if specified). If no objects match the criteria, the command returns an empty string.

You can use the **get_topological_constraints** command at the command prompt or nest it as an argument to another command (such as **report_topological_constraint**). You can also assign the **get_topological_constraints** result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example returns the topological constraints from the current_design

```
prompt> get_topological_constraints
{TPF_CONSTRAINT_0 TPF_CONSTRAINT_1 TPF_CONSTRAINT_3 TPF_CONSTRAINT_4}
```

The following example returns the topological constraints owned by net \'net1\'

```
prompt> get_topological_constraints -of_objects net1
{TPF_CONSTRAINT_1 TPF_CONSTRAINT_3}
```

The following example returns the topological constraints that have the 'start_sides' attribute set

```
prompt> get_topological_constraints -filter "start_sides!=\"\""
{TPF_CONSTRAINT_1 TPF_CONSTRAINT_4}
```

SEE ALSO

create_topological_constraint(2)
remove_topological_constraints(2)
report_topological_constraints(2)

get_topology_edges

Creates a collection by selecting topology edges from the current block.

SYNTAX

collection **get_topology_edges**
[-filter *expression*]
[-quiet]
[-design *design_name*]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns* | -of_objects *of_objects*]

Data Types

expression string
design_name string
exact_count integer
minimum_count integer
patterns list
of_objects list

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*, which must be a boolean expression based on the topology edge attributes. You can determine the topology edge attributes by using the **list_attributes** command. If the expression evaluates to **true**, the topology edge is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-design *design_name*

Specifies the design module in which to search for topology edges.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for topology edges level-by-level relative to the current instance. Use this option with *patterns* and not with the **-of_objects** option.

patterns

Matches the topology edge names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses asterisk (*) as the pattern.

-of_objects *of_objects*

Creates a collection containing the topology edges that are associated with the specified objects.

If the objects are topology nodes, the command returns the topology edges which the topology nodes are connected to, i.e. the set of topology edges which have the specified topology nodes as their start_node or end_node.

If the objects are topology plans, the command returns the topology edges owned by the plans.

If the objects are topology repeaters, the command returns the topology edges which own the repeaters.

If the objects are nets, supernets, or bundles, the command returns the topology edges that have one or more of those objects in their "objects" attribute collection.

If the objects are constraint groups, the command returns the topology edges which are members of the specified constraint groups.

Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

DESCRIPTION

This command creates a collection of topology edges that meet the selection criteria. It returns a collection handle if one or more topology edges meet the selection criteria. If no topology edges match the selection criteria, it returns an empty string.

You can use the **get_topology_edges** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of topology_edges by matching the name against the pattern:

```
prompt> get_topology_edges TOPOLOGY_EDGE*
{curplan/TOPOLOGY_EDGE0 curplan/TOPOLOGY_EDGE1 curplan/TOPOLOGY_EDGE2}
```

The following example creates a collection of all topology_edges:

```
prompt> get_topology_edges *
{curplan/TOPOLOGY_EDGE0 curplan/TOPOLOGY_EDGE1 curplan/TOPOLOGY_EDGE2 plan0/medge0 plan0/medge1}
```

The following example creates a collection of all topology_edges connected to topology_node "mynode":

```
prompt> get_topology_edges -of_objects [get_topology_nodes mynode]
{curplan/TOPOLOGY_EDGE1 curplan/TOPOLOGY_EDGE2}
```

SEE ALSO

create_topology_edge(2)
remove_topology_edges(2)
get_topology_nodes(2)
create_topology_node(2)
remove_topology_nodes(2)
report_topology_plans(2)

get_topology_nodes

Creates a collection by selecting topology nodes from the current block.

SYNTAX

collection **get_topology_nodes**
[-filter *expression*]
[-quiet]
[-design *design_name*]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns* | -of_objects *of_objects* | -of_ref_nodes *node_list*]

Data Types

expression string
design_name string
exact_count integer
minimum_count integer
patterns list
of_objects list
node_list list

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*, which must be a boolean expression based on the topology node attributes. You can determine the topology node attributes by using the **list_attributes** command. If the expression evaluates to **true**, the topology node is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-design *design_name*

Specifies the design module in which to search for topology nodes.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for topology nodes level-by-level relative to the current instance. Use this option with *patterns* and not with the **-of_objects** option.

patterns

Matches the topology node names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses asterisk (*) as the pattern.

-of_objects *of_objects*

Creates a collection containing the topology nodes that are associated with the specified objects.

If the objects are topology edges, the command returns the topology nodes which are connected as endpoints of the objects, i.e. the set of topology nodes which are either the start_node or end_node of the specified topology edges.

If the objects are topology_plans, the command returns the topology nodes owned by the plans.

Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

-of_ref_nodes *node_list*

Creates a collection topology nodes that reference the nodes in *node_list*.

DESCRIPTION

This command creates a collection of topology nodes that meet the selection criteria. It returns a collection handle if one or more topology nodes meet the selection criteria. If no topology nodes match the selection criteria, it returns an empty string.

You can use the **get_topology_nodes** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of topology_nodes by matching the name against the pattern:

```
prompt> get_topology_nodes TOPOLOGY_NODE*
{curplan/TOPOLOGY_NODE0 curplan/TOPOLOGY_NODE1 curplan/TOPOLOGY_NODE2}
```

The following example creates a collection of all topology_nodes:

```
prompt> get_topology_nodes *
{curplan/TOPOLOGY_NODE0 curplan/TOPOLOGY_NODE1 curplan/TOPOLOGY_NODE2 plan0/mnode0 plan0/mnode1}
```

The following example creates a collection of all topology_nodes that are endpoints of a collection of topology_edges:

```
prompt> get_topology_nodes -of_objects [get_topology_edges myedge*]
{curplan/TOPOLOGY_NODE1 curplan/TOPOLOGY_NODE2}
```

SEE ALSO

create_topology_node(2)
remove_topology_nodes(2)
get_topology_edges(2)
create_topology_edge(2)
remove_topology_edges(2)
report_topology_plans(2)

get_topology_plans

Creates a collection by selecting topology plans from the current block.

SYNTAX

collection **get_topology_plans**
[-filter *expression*]
[-quiet]
[-design *design_name*]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns* | -of_objects *of_objects*]

Data Types

expression string
design_name string
exact_count integer
minimum_count integer
patterns list
of_objects list

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*, which must be a boolean expression based on the topology plan attributes. You can determine the topology plan attributes by using the **list_attributes** command. If the expression evaluates to **true**, the topology plan is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-design *design_name*

Specifies the design module in which to search for topology plans.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for topology plans level-by-level relative to the current instance. Use this option with *patterns* and not with the **-of_objects** option.

patterns

Matches the topology plan names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses asterisk (*) as the pattern.

-of_objects *of_objects*

Creates a collection containing the topology plans that are associated with the specified objects.

If the objects are topology nodes or topology edges, the command returns the topology plans own those nodes and edges.

If the objects are topology_repeaters, the command returns the topology plans which own the edges that own the repeaters.

If the objects are nets, supernets, or bundles, the command returns the topology plans that have one or more of those objects in their "objects" attribute collection.

Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

DESCRIPTION

This command creates a collection of topology plans that meet the selection criteria. It returns a collection handle if one or more topology plans meet the selection criteria. If no topology plans match the selection criteria, it returns an empty string.

You can use the **get_topology_plans** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of topology_plans by matching the name against the pattern:

```
prompt> get_topology_plans TOPOLOGY_PLAN*
{TOPOLOGY_PLAN0 TOPOLOGY_PLAN1 TOPOLOGY_PLAN2}
```

The following example creates a collection of all topology_plans:

```
prompt> get_topology_plans *
{myplan0 myplan1 TOPOLOGY_PLAN0 TOPOLOGY_PLAN1 TOPOLOGY_PLAN2}.in -0.25in
```

SEE ALSO

create_topology_plan(2)
remove_topology_plans(2)
current_topology_plan(2)
report_topology_plans(2)

get_topology_repeaters

Creates a collection by selecting topology repeaters from the current block.

SYNTAX

collection **get_topology_repeaters**
[-filter *expression*]
[-quiet]
[-design *design_name*]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns* | -of_objects *of_objects*]

Data Types

expression string
design_name string
exact_count integer
minimum_count integer
patterns list
of_objects list

ARGUMENTS

-filter *expression*

Filters the collection with the value of *expression*, which must be a boolean expression based on the topology repeater attributes. You can determine the topology repeater attributes by using the **list_attributes** command. If the expression evaluates to **true**, the topology repeater is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-design *design_name*

Specifies the design module in which to search for topology repeaters.

-regexp

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing stops.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing stops.

-expect_each_pattern_matches

Checks that each pattern in *patterns* matches at least one object. If one or more patterns does not match, a Tcl error is raised and command processing stops.

-hierarchical

Searches for topology repeaters level-by-level relative to the current instance. Use this option with *patterns* and not with the **-of_objects** option.

patterns

Matches the topology repeater names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses asterisk (*) as the pattern.

-of_objects *of_objects*

Creates a collection containing the topology repeaters that are associated with the specified objects.

If the objects are topology edges, the command returns the topology repeaters which are owned by the edges.

If the objects are topology plans, the command returns the topology repeaters owned by the topology edges owned by the plans.

Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

DESCRIPTION

This command creates a collection of topology repeaters that meet the selection criteria. It returns a collection handle if one or more topology repeaters meet the selection criteria. If no topology repeaters match the selection criteria, it returns an empty string.

You can use the **get_topology_repeaters** command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example creates a collection of topology_repeaters by matching the name against the pattern:

```
prompt> get_topology_repeaters TOPOLOGY_REPEATER*
{curplan/edge0/TOPOLOGY_REPEATER0 curplan/edge0/TOPOLOGY_REPEATER1
curplan/edge0/TOPOLOGY_REPEATER2}
```

SEE ALSO

create_topology_repeater(2)
remove_topology_repeaters(2)
get_topology_edges(2)
get_topology_plans(2)
report_topology_plans(2)

get_trace_option

Get the current option value controlling behavior of command tracing and output annotation..

SYNTAX

get_trace_option [-command name
|-profile
|-memory_threshold
|-cpu_threshold
|-time_treshold]
[-annotate | -is_traced]

string *name*

ARGUMENTS

-command *name*

This option is mutually exclusive with -profile, -memory_threshold, -cpu_threshold, and -time_threshold. The option identifies the command for which tracing or annotation option is returned.

-annotate

If this option is given, then the *annotate* setting for the given command is returned. This option is mutually exclusive with -is_traced and is meaningful only in combination with the -command option.

-is_traced

If this option is given, then returns 1 if the given command is traced and logged, 0 otherwise. This option is mutually exclusive with -annotate and is meaningful only in combination with the -command option.

-profile

This option is mutually exclusive with all others. The option returns a list of the currently enabled profile metrics, or "all" if all metrics are enabled.

-memory_threshold

This option is mutually exclusive with all others. The option returns the currently set memory profile threshold, if any, in kilobytes. If no threshold is set, then it returns the string "unset".

-cpu_threshold

This option is mutually exclusive with all others. The option returns the currently set cpu profile threshold, if any, in seconds. If no threshold is set, then it returns the string "unset".

-time_threshold

This option is mutually exclusive with all others. The option returns the currently set wall clock time profile threshold, if any, in kilobytes. If no threshold is set, then it returns the string "unset".

DESCRIPTION

The command is used to query the current setting for the command annotation option that is set with the set_trace_option command, or profile metric settings.

EXAMPLES

The following example sets the annoation type of the **get_attribute** command.

```
prompt> get_trace_option -command get_attribute -annotate
annotate
```

The following example sets, then gets, the currently enabled profile metrics.

```
prompt> set_trace_topion -profile {cpu time}
prompt> get_trace_topion -profile
cpu time
```

The following example gets the threshold for memory profile metric before one has been set.

```
prompt> get_trace_option -memory_threshold
unset
```

The following example will list the commands in the Builtins command group that are not traced and logged.

```
prompt> foreach cmd [dict get [get_defined_commands -details -groups Builtins] commands] {
?  if {![get_trace_option -command $cmd -is_traced]}
?    {echo $cmd}
?  }
```

SEE ALSO

log_trace(2)
annotate_trace(2)
set_trace_option(2)

get_tracks

Creates a collection of tracks from the current design.

SYNTAX

collection **get_tracks**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count integer
minimum_count integer
patterns string
objects list
point list
region list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on bound attributes. You can determine the shape attributes by using the **list_attributes** command. If the expression evaluates to **true**, the shape is included in

the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for tracks level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-of_objects *of_objects*

Creates a collection of tracks that are located on the specified layers or in the specified edit groups. Each object in the list can be a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

patterns

Matches the track names in the current design against the specified patterns. You can specify the patterns by using the following formats:

- A collection of tracks
- An asterisk (*), which indicates all tracks
- Track names, which are in the format TRACK_*object_id*

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one

except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all tracks at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all tracks that are completely inside the specified region and doesn't include tracks which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all tracks that are completely inside the specified region and the tracks which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all tracks which are abut/touching from inside or outside to the specified region and the tracks whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input

polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command returns a collection of tracks of the current design that meet the selection criteria. You can use this command to query the tracks inside the database by layer, by point, or by region. You can also use this command to query tracks by filtering their attributes.

EXAMPLES

The following example returns the tracks that are located on the METAL2 layer.

```
prompt> get_tracks * -filter "layer~=METAL2"
{"TRACK_4683"}
```

The following example returns all tracks.

```
prompt> get_tracks "TRACK_*"
{"TRACK_4683 TRACK_5389"}
```

The following examples shows how to use the **get_attribute** command to query the value of a track attribute.

```
prompt> get_attribute [get_tracks  TRACK_43] layer_name
METAL2
prompt> get_attribute [get_tracks  TRACK_43] layer
{"METAL"}

prompt> get_attribute [get_tracks TRACK_43] bbox
{{0.000 0.000} {100.000 100.000}}

prompt> get_attribute [get_tracks TRACK_43] start
0.000 0.000
```

To get a complete report of the attribute values of all tracks, use the **report_attributes** command.

SEE ALSO

get_attribute(2)

report_attributes(2)

get_undo_info

Returns information about the undo history and the undo system.

SYNTAX

array **get_undo_info**
[-name *marker_name* | -current | -system | -user | -all | -command *command_name*]
[-details]

Data Types

marker_name string
command_name string

ARGUMENTS

-name *marker_name*

Return information about the undo marker with the specified user or system name.

-current

Return information about the current undo marker. If the system is at its most recent state, then no marker is the current marker and an empty string will be returned.

-system

Return a list of all system-generated markers in the undo history.

-user

Return a list of all user-generated markers in the undo history.

-all

Return a list of all markers in the undo history.

-command *command_name*

Return information about the undoability of the specified command.

-details

When used with the *-all*, *-user*, or *-system*, include detailed information about each marker in the returned marker list.

DESCRIPTION

This command returns information about the undo system, the undo history, and the undoability of commands. If the *-name* or *-current* option is used, an array of information about the specified marker is returned. The *-name* option specifies the marker with user or system name *marker_name*. The *-current* option specifies the marker corresponding to the current point in undo history, being the point before the command represented by the marker in the case of system markers. If system is at its most recent state, then no marker is the current marker, and an empty string will be returned.

If the *-all*, *-user*, or *-system* option is used, then a list of names of all, user-generated, or system-generated markers will be returned, respectively. If the *-details* option is also used, then a list of arrays of complete information about the specified markers is returned.

The following attribute/value pairs are included in marker information arrays (note: user markers are created with the *create_undo_marker* command; system markers are automatically created whenever an undoable command is executed):

marker	System name - always non-empty
name	User name - non-empty for user markers
command_name	Name of the command this marker corresponds to - non-empty for system markers
memory	Memory used by the marker to store data required to transition system state to or from the next more recent marker or the latest state if there are no more recent markers.
current	True if this is the current marker, false otherwise.

If the *-command* option is used, this command returns 0, 1, or 2 if the command is not undoable, undoable, or unknown to the undo system, respectively.

If no options are specified, then this command returns an array of information about the undo system.

EXAMPLES

The following gets a detailed list of all undo markers:

```
prompt> get_undo_info -all -details
{marker "SNPS_MARKER_131923" name "my_marker" command_name "" memory 0 current false}
{marker "SNPS_MARKER_131924" name "" command_name "create_edit_group" memory 19932 current false}
{marker "SNPS_MARKER_131925" name "" command_name "add_to_edit_group" memory 20416 current false}
```

The following gets a list of all system markers:

```
prompt> get_undo_info -system
"SNPS_MARKER_131924" "SNPS_MARKER_131925"
```

The following gets a list of all user markers:

```
prompt> get_undo_info -user
"my_marker"
```

The following gets the undoability of the *place_and_optimize_design* command:

```
prompt> get_undo_info -command place_and_optimize_design
Information: The command 'place_and_optimize_design' is not undoable. (UNDO-015)
0
```

SEE ALSO

undo(2)
redo(2)
create_undo_marker(2)
eval_with_undo(2)

get_unix_variable

This is a synonym for the **getenv** command.

SEE ALSO

gettenv(2)
printenv(2)
printvar(2)
set(2)
setenv(2)
sh(2)
unset(2)

get_user_units

This command returns a string representing the input or output unit for one quantity type.

SYNTAX

string **get_user_units**
  -input | -output
  -type *unit_type*
  [-numeric]

string *unit_type*

ARGUMENTS

-input

Get the unit for data input. Either **-input** or **-output** must be specified.

-output

Get the unit for data output. Either **-input** or **-output** must be specified.

-type *unit_type*

Specifies the type of quantity, which must be one of "time", "resistance", "capacitance", "voltage", "power" or "current".

-numeric

If specified, return the value as a number representing the unit value in terms of the base unit for that quantity type. Base units are seconds, Ohms, Farads, Volts, and Amperes.

DESCRIPTION

This command returns a string representing the input unit for one quantity type. The tool maintains separate units for input and input. There are input unit values for time, resistance, capacitance, power, voltage and current.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example shows how to get the input unit for time. With the **-numeric** option, the value is returned in terms of the base unit for that quantity type (seconds, for time).

```
prompt> get_user_units -input -type time
1.00ps
prompt> get_user_units -input -type time -numeric
1e-12
```

SEE ALSO

report_user_units(2)
set_user_units(2)

get_utilization_configurations

Creates a collection of utilization configurations from the scopes lib, tech and block.

SYNTAX

collection **get_utilization_configurations**
[-block *block*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect exact_count]
[-expect_at_least minimum_count]
[-expect_each_pattern_matches]
[-scope scope]
[patterns]

Data Types

expression string
patterns list
scope string

DESCRIPTION

This command creates and returns a collection of all the existing utilization configurations.

ARGUMENTS

-block *block*

Specifies the block in which the configurations are to be found.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the utilization-configuration attributes. You can determine the utilization-configuration attributes by using the **list_attributes** command. If the expression evaluates to **true**, the specific configuration is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-scope *scope*

The values it can take are 'lib', 'tech' and 'block'. By default, the order of search is block, lib and then tech i.e. the configuration is first searched in the current block, then the current library and finally the tech associated with the current library.

patterns

Matches the configuration names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

EXAMPLES

The following example lists the configurations created through the **create_utilization_configuration** command.

```
prompt> create_utilization_configuration config_lib -capacity core_area -scope lib
prompt> create_utilization_configuration config_tech -capacity boundary -scope tech
prompt> create_utilization_configuration config_block -capacity site_array -scope block

prompt> get_utilization_configurations
{config_lib config_tech config_block}
prompt> get_utilization_configurations -scope lib
{config_lib}
prompt> get_utilization_configurations -scope tech
{config_tech}
prompt> get_utilization_configurations -scope block
```

{config_block}

SEE ALSO

create_utilization_configuration(2)
remove_utilization_configurations(2)
report_utilization(2)

get_via_defs

Creates a collection by selecting via_defs from a block or tech.

SYNTAX

collection **get_via_defs**
[-design *design*]
[-library *library*]
[-tech *tech*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
patterns | -of_objects *of_objects*

Data Types

design collection
library collection
tech string
expression string
exact_count int
minimum_count int
patterns string
of_objects list

ARGUMENTS

-design *design*

Specifies the block for finding via_defs. If neither a design nor tech is specified, the current block is used.

-library *library*

Specifies the tech for finding via_defs. The tech contained by or referenced by the given library is used. If neither a design nor tech is specified, the current block is used.

-tech *tech*

Specifies the tech for finding via_defs. If neither a design nor tech is specified, the current block is used.

-filter *expression*

Filters the collection with *expression*. For any via_def that matches the patterns option, the expression is evaluated based on the via_def's attributes. If the expression evaluates to *true*, the via_def is included in the result. By default, this option is off.

Use the **list_attributes** command to determine the via_def attributes.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive. **-nocase** and **-exact** are mutually exclusive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *of_objects*

Creates a collection containing the via_defs of the specified vias. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches the via_def names against the patterns in the specified block or tech.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

DESCRIPTION

This command creates a collection of via_defs by selecting via_defs from the specified block or tech that meet the selection criteria. It returns a collection handle if one or more via_defs meet the selection criteria. If no via_defs match the selection criteria, it returns

an empty string.

Use the **get_via_defs** command as an argument to another command or assign its result to a variable. Refer to the example below for more information.

See the collection command man page for information about working with collections.

EXAMPLES

The following example creates the collection of all custom via_defs used in the current block:

```
prompt> get_via_defs -of_objects VIA_C*
{FATVIA12_230_530 FATVIA12_530_230}
```

The following example creates the collection of all via_defs from the tech of library 'test' with cut layer VIA1 :

```
prompt> get_via_defs -tech [get_techs -of_objects test] -filter "cut_layer_name == VIA1"
{VIA12 FATVIA12}
```

SEE ALSO

get_attribute(2)
query_objects(2)
report_via_defs(2)

get_via_ladders

Creates a collection by selecting via ladder from the design.

SYNTAX

collection **get_via_ladders**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*]

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns string
objects list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with *expression*. For any via ladder that match the patterns option, the expression is evaluated based on the attributes of the via ladder. If the expression evaluates to *true*, the via ladder is included in the result. By default, this option is off.

Use the **list_attributes** command to determine the via ladder attributes.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for vias level-by-level relative to the current instance. This option is useful only with *patterns* and not with **-of_objects**.

-of_objects *of_objects*

Creates a collection containing the via ladders connected to the specified net. Each object in the list is a name, pattern, or collection.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

patterns

Matches the via ladder names against the patterns in the current design. You can specify the patterns with the following formats:

- * (asterisk) indicates all via ladders

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

DESCRIPTION

This command creates a collection of via ladders by selecting via ladders from the current design that meet the selection criteria. It returns a collection handle if one or more via ladders meet the selection criteria. If no via ladders match the selection criteria, it returns an empty string.

Use the **get_via_ladders** command as an argument to another command or assign its result to a variable. Refer to the example

below for more information.

See the collection command man page for information about working with collections.

EXAMPLES

The following example create via ladder collections of net MemData[31]:

```
prompt> get_via_ladders -of_objects MemData[31]
{VIA_LADDER_0}
```

The following example gets all via ladders using '*' :

```
prompt> get_via_ladders *
{VIA_LADDER_0 VIA_LADDER_1}
```

The following example gets via ladders using regular expression match :

```
prompt> get_via_ladders -regexp VIA_LADDER_.
{VIA_LADDER_0 VIA_LADDER_1}
```

SEE ALSO

get_attribute(2)
report_attributes(2)

get_via_matrixes

Creates a collection by selecting via matrixes from the design.

SYNTAX

collection **get_via_matrixes**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*]
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns string
objects list
point list
region list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with *expression*. For any via matrix that match the patterns option, the expression is evaluated based on the via matrix's attributes. If the expression evaluates to *true*, the via matrix is included in the result. By default, this option is off.

Use the **list_attributes** command to determine the via matrix attributes.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for vias level-by-level relative to the current instance. This option is useful only with *patterns* and not with **-of_objects**.

-of_objects *of_objects*

Creates a collection containing the via matrixes connected to the specified net, or edit groups. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

patterns

Matches the via matrix names against the patterns in the current design. You can specify the patterns with the following formats:

- * (asterisk) indicates all via matrixes

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all via_matrices at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

In addition, you cannot use the **-hierarchical** option with the **-at** option.

-within *region*

Creates a collection that contains all via_matrices that are completely inside the specified region and do not overlap the boundary. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

In addition, you cannot use the **-hierarchical** option with the **-within** option.

-touching *region*

Creates a collection that contains all via_matrices that are inside the specified region, including those that overlap the boundary. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

In addition, you cannot use the **-hierarchical** option with the **-touching** option.

-intersect *region*

Creates a collection that contains all via_matrices that intersect the boundary of the specified region and at least part of the via_matrix is outside of the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} or {*llx lly urx ury*}, which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

In addition, you cannot use the **-hierarchical** option with the **-intersect** option.

DESCRIPTION

This command creates a collection of via matrixes by selecting via matrixes from the current design that meet the selection criteria. It returns a collection handle if one or more via matrixes meet the selection criteria. If no via matrixes match the selection criteria, it returns an empty string.

Use the **get_via_matrixes** command as an argument to another command or assign its result to a variable. Refer to the example below for more information.

See the collection command man page for information about working with collections.

EXAMPLES

The following examples create via matrix collections of net MemData[31]:

```
prompt> get_via_matrixes -of_objects MemData[31]
{VIA_MATRIX_0}
```

The following examples get all via matrixes for the same via definition VIA12 :

```
prompt> get_via_matrixes -filter "via_def_name==VIA12"
{VIA_MATRIX_0 VIA_MATRIX_1}
```

SEE ALSO

get_attribute(2)
query_objects(2)

get_via_regions

Creates a collection of via regions in the frame block that match the specified criteria.

SYNTAX

collection **get_via_regions**
[-design *block*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
patterns
| -of_objects *of_objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

block collection
expression string
exact_count int
minimum_count int
patterns string
of_objects list
region list
point list

ARGUMENTS

-design *block*

Specifies the block for finding via_regions. If no design is specified, the current block is used.

-filter *expression*

Filters the collection with *expression*. For any via_regions that match the patterns option, the expression is evaluated based on the attributes for the via_region. If the expression evaluates to *true*, the via_region is included in the result. By default, this option is off.

Use the **list_attributes** command to determine the via_region attributes.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a real regular expression rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matching case-insensitive. **-nocase** and **-exact** are mutually exclusive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies the expected number of objects to find. If the command finds a different number of objects, a Tcl error is raised and command processing terminates.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error is raised and command processing terminates.

-expect_each_pattern_matches

Forces each pattern in *patterns* to match at least one object. If one or more patterns does not match, a Tcl error is raised and command processing terminates.

-of_objects *of_objects*

Creates a collection containing the via_regions of the specified terminals. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

patterns

Matches the via_region names against the patterns in the specified block.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all via_regions at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-within *region*

Creates a collection that contains all via_regions that are completely inside the specified region and doesn't include via_regions which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

-touching *region*

Creates a collection that contains all via_regions that are completely inside the specified region and the via_regions which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-intersect *region*

Creates a collection that contains all via_regions which are abut/touching from inside or outside to the specified region and the via_regions whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

DESCRIPTION

This command creates a collection of via_regions by selecting via_regions from the specified block that meet the selection criteria. It returns a collection handle if one or more via_regions meet the selection criteria. If no via_regions match the selection criteria, it returns an empty string.

Use the **get_via_regions** command as an argument to another command or assign its result to a variable. Refer to the example below for more information.

See the collection command man page for information about working with collections.

EXAMPLES

The following example creates the collection of all via_regions of terminal Reset in current block:

```
prompt> get_via_regions -of_objects Reset
{Reset/VR_0 Reset/VR_1}
```

The following example creates the collection of all system via_regions in current block:

```
prompt> get_via_regions * -filter "type == SYSTEM"
{D1/VR_0  D1/VR_1}
```

SEE ALSO

create_via_region(2)
get_attribute(2)
query_objects(2)
report_via_regions(2)

get_via_rules

Returns a collection by selecting via_rules from a block and tech.

SYNTAX

collection **get_via_rules**
[-design *design*]
[-library *library*]
[-tech *tech_object*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns* | -of_objects *of_objects*]

Data Types

design collection
library collection
tech_object collection
expression string
exact_count integer
minimum_count integer
patterns string
of_objects list

ARGUMENTS

-design *design*

Specifies the block for finding via_rules. If neither a design nor tech is specified, the current block is used. If current block is not set then current lib is used.

-library *library*

Specifies the tech for finding via_rules. The tech contained by or referenced by the given library is used. If neither a design nor tech is specified, the current block is used. If current block is not set then current lib is used.

-tech *tech_object*

Specifies the tech for finding via_rules. If neither a design nor tech is specified, the current block is used. If current block is not set

then current lib is used.

-filter *expression*

Filters the collection with *expression*. For any via rule that match the patterns option, the expression is evaluated based on the via rule's attributes. If the expression evaluates to *true*, the via rule is included in the result.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-of_objects *of_objects*

Creates a collection containing the via rules of the specified techs. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches via rule names against the *patterns* argument. The *patterns* argument can include the wildcard character "*".

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

DESCRIPTION

The **get_via_rules** command creates a collection of via rules from the specified block scope and the tech scope that match certain criteria. The command returns a collection of via rules if any via rule matches the criteria. If no objects match the criteria, the empty

string is returned. By default, the command returns the via rules in both block and tech scope. If the search is to be limited to either block or tech scope, the same can be specified as an option.

You can use the **get_via_rules** command at the command prompt, or you can nest it as an argument to another command, such as **report_attributes**. In addition, you can assign the **get_via_rules** result to a variable.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example returns a collection that contains all the via rules in the block and tech scope of the current block.

```
prompt> get_via_rules
{VR1 VL1 VL1}
```

SEE ALSO

collections(2)
query_objects(2)
shell.common.collection_result_display_limit(3)

get_vias

Creates a collection by selecting vias from the design.

SYNTAX

collection **get_vias**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[-include_shield]
[-shield_only]
[-include_lib_cell]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design collection
expression string
exact_count integer
minimum_count integer
patterns string
objects list
point list
region list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with *expression*. For any via that match the patterns option, the expression is evaluated based on the via's attributes. If the expression evaluates to *true*, the via is included in the result. By default, this option is off.

Use the **list_attributes** command to determine the via attributes.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for vias level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-include_shield

Specifies to include the shielding vias when -of_objects contains nets. It is only valid when -of_objects contains nets.

This option is mutually exclusive with -shield_only.

-shield_only

Specifies to get the shielding vias only when -of_objects contains nets. It is only valid when -of_objects contains nets.

This option is mutually exclusive with -include_shield.

-include_lib_cell

Search for vias in the **lib_cell**. By default, **lib_cell** vias are excluded from search.

This option must be used with -hierarchical option.

-of_objects *of_objects*

Creates a collection containing the vias connected to the specified net, port, terminal, lib_pin, or edit groups. Each object in the list is a name, pattern, or collection.

When ports are specified by **-of_objects**, **get_vias** searches for vias attached to the terminals associated with the specified ports.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

patterns

Matches the via names against the patterns in the current design. You can specify the patterns with the following formats:

- * (asterisk) indicates all vias
- *VIA** indicates all vias
- *VIA_S** indicates all simple vias
- *VIA_C** indicates all custom vias
- *VIA_C_120* indicates one simple via.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all vias at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all vias that are completely inside the specified region and doesn't include vias which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one

except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching ***region***

Creates a collection that contains all vias that are completely inside the specified region and the vias which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect ***region***

Creates a collection that contains all vias which are abut/touching from inside or outside to the specified region and the vias whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of vias by selecting vias from the current design that meet the selection criteria. It returns a collection handle if one or more vias meet the selection criteria. If no vias match the selection criteria, it returns an empty string.

Use the **get_vias** command as an argument to another command or assign its result to a variable. Refer to the example below for more information.

See the collection command man page for information about working with collections.

EXAMPLES

The following examples create via collections of net n300:

```
prompt> get_vias -of_objects n300
{"VIA_C_67"}
```

The following examples get all vias from the same via definition VIA12 :

```
prompt> get_vias -filter "via_def_name==VIA12"
{"VIA_S_120", "VIA_S_122", "VIA_S_123"}
```

SEE ALSO

get_attribute(2)
query_objects(2)

get_view_switch_list

Returns the value of the specified design, library, or global view switch list.

SYNTAX

string **get_view_switch_list**
[-design *design* | -library *library* | -global]
[-explicit]

Data Types

design string
library string

ARGUMENTS

-design *design*

Specifies the design for which to return the view switch list. This option is mutually exclusive with the **-library** and **-global** options.

-library *library*

Specifies the library for which to return the view switch list. This option is mutually exclusive with the **-design** and **-global** options.

-global

Returns the global, session-wide view switch list. This option is mutually exclusive with the **-design** and **-library** options.

-explicit

Returns the view switch list that was explicit set on the specified entity. If this option is not specified, the command returns the effective view switch list that is applied to the entity in the case where no view switch list was explicitly set.

DESCRIPTION

This command returns the effective view switch list of a design, library, or global session.

The view switch list is a precedence-ordered list of design views that is applied when attempting to bind a design. If you explicitly set the switch list for the design, then that view switch list is used. If not explicitly set, the command returns the view switch list for the design's owning library. If the owning library does not have a view switch list that was explicitly set, then the global view switch list is used. The view switch list attribute is persistent for designs and libraries, but not for the global session.

It is an error to specify more than one of the **-global**, **-library**, or **-design** options. If none of these options are specified, the command returns the global view switch list. An empty view switch list is displayed as "{ }". If you specified the **-explicit** option and the command returns an empty view switch list, then no explicitly set view switch list exists on the entity. If you omitted the **-explicit** option and the command returns an empty view switch list, then the view switch list is empty for this entity and its owners. If the entity is a design in this case, then it cannot be bound.

EXAMPLES

The following example gets the effective view switch list of design "mydesign":

```
prompt> get_view_switch_list -design mydesign
abstract outline
```

The following example gets the effective view switch list of library "mylib":

```
prompt> get_view_switch_list -library mylib
frame abstract design
```

The following examples each get the global view switch list:

```
prompt> get_view_switch_list
design frame abstract outline
```

```
prompt> get_view_switch_list -global
design frame abstract outline
```

SEE ALSO

set_view_switch_list(2)
change_view(2)

get_virtual_connections

Gets a list of virtual connection that match the specified input criteria.

SYNTAX

status **get_virtual_connections**
patterns
| -of_objects *objects*

Data Types

patterns list
objects list

ARGUMENTS

patterns

Matches virtual connection names against patterns. Patterns can include the wildcard characters "*" and "?". Patterns can also include collections of type virtual connection.

-of_objects *objects*

The command returns a list of virtual connection connected to the specified objects. Each object can be a pin, port and cell.

DESCRIPTION

This command gets a list of virtual connection that match the specified input criteria.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples get the virtual connection.

prompt> **get_virtual_connections SNPS_VC_1**

```
prompt> get_virtual_connections {SNPS_VC_1 SNPS_VC_2}

prompt> get_virtual_connections VC*

prompt> get_virtual_connections \
    -of_objects [get_pins {U212/ZN U256/I}]

prompt> get_virtual_connections \
    -of_objects [get_cells {U300 U391}]

prompt> get_virtual_connections \
    -of_objects [get_ports WriteAdd[0]]

prompt> get_virtual_connections \
    -of_objects {U300/ZN U212 MemData[0]}
```

SEE ALSO

add_pins_to_virtual_connection(2)
create_virtual_connection(2)
get_attribute(2)
place_eco_cells(2)
remove_pins_from_virtual_connection(2)
remove_virtual_connections(2)
set_attribute(2)

get_voltage_area_rules

Returns a collection of voltage_area_rules from the design.

SYNTAX

collection **get_voltage_area_rules**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[*patterns*]

Data Types

design collection
expression string
exact_count int
minimum_count int
patterns string

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a boolean expression based on the voltage_area_rule attributes. You can determine the voltage_area_rule attributes by using the **list_attributes** command. If the expression evaluates to **true**, the voltage_area_rule is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are

mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. Wildcards are considered as plain characters. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options.

patterns

Matches voltage_area_rule names against *patterns*. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the pattern.

DESCRIPTION

This command creates a collection of voltage_area_rules from the design. The command returns a collection handle (identifier) if any voltage_area_rules match the value of the *patterns* option and pass the filter (if specified). If no objects match the criteria, the command returns an empty string.

You can use the **get_voltage_area_rules** command at the command prompt or nest it as an argument to another command (such as **report_voltage_area_rules**). You can also assign the **get_voltage_area_rules** result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following example returns a collection of voltage_area_rules with names starting with "RULE".

prompt> **get_voltage_area_rules RULE***

The following example returns a collection of voltage_area_rules which allow physical feedthrough.

```
prompt> get_voltage_area_rules -filter is_physical_feedthrough_allowed
```

SEE ALSO

create_voltage_area_rule(2)
remove_voltage_area_rules(2)
report_voltage_area_rules(2)

get_voltage_area_shapes

Creates a collection of voltage_area shapes from the current design.

SYNTAX

collection **get_voltage_area_shapes**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design string
expression string
exact_count integer
minimum_count integer
patterns list
objects list
point list
region list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the voltage_area shapes attributes. You can determine the voltage_area shapes attributes by using the **list_attributes** command. If the expression

evaluates to **true**, the voltage area shapes are included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for voltage area shapes level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-of_objects *objects*

Creates a collection containing the voltage_area shapes of the specified voltage_areas, power_domains, cells, or edit groups. Each object in the list is a name, pattern, or collection.

The *patterns* argument and **-of_objects** option are mutually exclusive. You can specify only one of these options. If you do not specify either option, the command uses * (asterisk) as the *pattern*.

patterns

Matches the voltage_area shape names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all voltage area shapes at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all voltage area shapes that are completely inside the specified region and doesn't include voltage area shapes which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all voltage area shapes that are completely inside the specified region and the voltage area shapes which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all voltage area shapes which are abut/touching from inside or outside to the specified region and the voltage area shapes whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses *

(asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of voltage_area shapes from the current design that match certain criteria. The command returns a collection handle (identifier) if any voltage_area shapes match the value of the *patterns* option and pass the filter (if specified). If no objects match the criteria, the command returns an empty string.

You can use the **get_voltage_area_shapes** command at the command prompt or nest it as an argument to another command (such as **remove_voltage_area_shapes**). You can also assign the **get_voltage_area_shapes** result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following returns voltage_area shapes of a voltage_area

```
prompt> get_voltage_area_shapes -of_objects [get_voltage_area VA_1]
{VA1}
```

SEE ALSO

create_voltage_area(2)
create_voltage_area_shape(2)
get_voltage_areas(2)
remove_voltage_area_shapes(2)
remove_voltage_areas(2)
report_voltage_areas(2)

get_voltage_areas

Creates a collection of voltage_areas from the current design.

SYNTAX

collection **get_voltage_areas**
[-design *design*]
[-filter *expression*]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect *exact_count*]
[-expect_at_least *minimum_count*]
[-expect_each_pattern_matches]
[-hierarchical]
[*patterns*
| -of_objects *objects*
| -at *point*
| -within *region*
| -touching *region*
| -intersect *region*]

Data Types

design string
expression string
exact_count integer
minimum_count integer
patterns list
objects list
point list
region list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-filter *expression*

Filters the collection with the value of *expression*, which must be a Boolean expression based on the voltage_area attributes. You

can determine the voltage_area attributes by using the **list_attributes** command. If the expression evaluates to **true**, the voltage_area is included in the collection. By default, this option is off.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. **-regexp** and **-exact** are mutually exclusive.

-nocase

Makes matches case-insensitive.

-exact

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. **-exact** and **-regexp** are mutually exclusive.

-expect *exact_count*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

-expect_at_least *minimum_count*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

-expect_each_pattern_matches

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

-hierarchical

Searches for voltage areas level-by-level relative to the current instance. This option is useful with *patterns*, **-within**,**-touching**, **-intersect** and **-at**, but not with **-of_objects**.

-of_objects *objects*

Creates a collection containing the voltage_areas associated with the specified objects. The objects can be voltage_area shapes, power domains, supply nets, cells, or edit groups. Each object in the list is a name, pattern, or collection.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

patterns

Matches the voltage_area names against the patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the **-regexp** option.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

-at *point*

Creates a collection that contains all voltage_areas at the specified point. The format for specifying a point is {*x y*}.

The coordinate unit is specified in the technology file; typically it is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-within *region*

Creates a collection that contains all voltage_areas that are completely inside the specified region and doesn't include voltage_areas which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-touching *region*

Creates a collection that contains all voltage_areas that are completely inside the specified region and the voltage_areas which are abut/touching from inside with the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

-intersect *region*

Creates a collection that contains all voltage_areas which are abut/touching from inside or outside to the specified region and the voltage_areas whose part of it is outside the specified region. The region boundary can be rectangle or polygon.

The format for specifying a rectangle is {{*llx lly*} {*urx ury*}} , which specifies the lower-left and upper-right corners of the rectangle.

The format for specifying a polygon is {{*x1 y1*} {*x2 y2*} ... {*xN yN*} {*x1 y1*}}, where each {*x y*} pair specifies one point of the input polygon. A valid polygon must be rectilinear, so the startpoint and endpoint of the polygon are the same point.

The coordinate unit is specified in the technology file; typically the unit is microns.

The *patterns*, **-of_objects**, **-within**, **-touching**, **-intersect**, and **-at** arguments are mutually exclusive; you can specify only one

except **-within** and **-intersect** can be specified together. If you do not specify any of these arguments, the command uses * (asterisk) as the *pattern*.

By default, query will be limited to current physical hierarchy. **-hierarchical** can be specified to query objects across physical hierarchies.

DESCRIPTION

This command creates a collection of voltage_areas from the current design that match certain criteria. The command returns a collection handle (identifier) if any voltage_areas match the value of the *patterns* option and pass the filter (if specified). If no objects match the criteria, the command returns an empty string.

You can use the **get_voltage_areas** command at the command prompt or nest it as an argument to another command (such as **report_voltage_areas**). You can also assign the **get_voltage_areas** result to a variable.

See the **collections** command man page for information about working with collections.

EXAMPLES

The following returns voltage_area from its shapes

```
prompt> get_voltage_areas -of_objects [get_voltage_area_shapes VOLTAGE_AREA_SHAPE_1]
{VA_1}
```

SEE ALSO

create_voltage_area(2)
create_voltage_area_shape(2)
get_voltage_area_shapes(2)
remove_voltage_area_shapes(2)
remove_voltage_areas(2)
report_voltage_areas(2)

get_vsdc

Get the name of the current VSDC file.

SYNTAX

boolean **get_vsdc**

DESCRIPTION

This command prints the name of the VSDC file that is currently open for recording VSDC guidance for Formality. If no VSDC file has been specified using the **set_vsdc** command, a message is printed.

EXAMPLES

In this example, we query the name of the VSDC file and find out that setup information is not being recorded yet. Then we specify the file name and repeat the query.

```
prompt> get_vsdc
VSDC guidance is not enabled.
1
prompt> set_vsdc cache_ctrl.vsdc
1
prompt> get_vsdc
The current VSDC file is /project/chip/cache_ctrl/cache_ctrl.vsdc.
1
```

SEE ALSO

set_vsdc(2)
set_svf(2)
Formality User Guide

get_working_design_stack

Gets the current working design stack.

SYNTAX

collection **get_working_design_stack**
[-instance]

ARGUMENTS

-instance

When specified the command returns pushed instance of current block within top hierarchy

DESCRIPTION

The command returns a collection that contains the current working design stack or the pushed instance of current block within top hierarchy

The first object of the returned collection is the top-level design, the second is the child design of the top-level design, and so on. The last element of the returned collection is the current working design, which you can get by using the **current_design** command.

EXAMPLES

The following example uses the **get_working_design_stack** command to return the working design stack before and after running the **set_working_design** command.

```
prompt> get_working_design_stack
{ORCA}
prompt> set_working_design -push I_ORCA_TOP/I_BLENDER_4
1
prompt> get_working_design_stack
{ORCA BLENDER_2}
prompt> get_working_design_stack -instance
{I_ORCA_TOP/ORCA BLENDER_2}
```

SEE ALSO

current_design(2)
set_working_design(2)
set_working_design_stack(2)

getenv

Returns the value of a system environment variable.

SYNTAX

string **getenv**
variable_name

Data Types

variable_name string

ARGUMENTS

variable_name

Specifies the name of the environment variable to be retrieved.

DESCRIPTION

The **getenv** command searches the system environment for the specified *variable_name* and sets the result of the command to the value of the environment variable. If the variable is not defined in the environment, the command returns a Tcl error. The command is catchable.

Environment variables are stored in the **env** Tcl array variable. The **getenv**, **setenv**, and **printenv** environment commands are convenience functions to interact with this array.

The application you are running inherited the initial values for environment variables from its parent process (that is, the shell from which you invoked the application). If you set the variable to a new value using the **setenv** command, you see the new value within the application and within any new child processes you initiate from the application using the **exec** command. However, these new values are not exported to the parent process. Further, if you set an environment variable using the appropriate system command in a shell you invoke using the **exec** command, that value is not reflected in the current application.

See the **set**, **unset**, and **printvar** commands for information about working with non-environment variables.

EXAMPLES

In the following example, **getenv** returns you to your home directory:

```
prompt> set home [getenv "HOME"]
/users/disk1/bill

prompt> cd $home

prompt> pwd
/users/disk1/bill
```

In the following example, **setenv** changes the value of an environment variable:

```
prompt> getenv PRINTER
laser1

prompt> setenv PRINTER "laser3"
laser3

prompt> getenv PRINTER
laser3
```

In the following example, the requested environment variable is not defined. The error message shows that the Tcl variable **env** was indexed with the value UNDEFINED, which resulted in an error. In the second command, **catch** is used to suppress the message.

```
prompt> getenv "UNDEFINED"
Error: can't read "env(UNDEFINED)": no such element in array
       Use error_info for more info. (CMD-013)

prompt> if {[catch {getenv "UNDEFINED"} msg]} {
         setenv UNDEFINED 1
        }
```

SEE ALSO

catch(2)
exec(2)
printenv(2)
printvar(2)
set(2)
setenv(2)
unsetenv(2)
unset(2)

group_cells

Creates a new level of hierarchy

SYNTAX

collection **group_cells**
[*cell_list*]
[-module_name *module_name*]
[-cell_name *cell_name*]
[-phys_wrapper]
[-hdl_block *block_name*]
[-hdl_all_blocks]
[-hdl_bussed]
[-enable_mim *mim_module_name*]
[-mim_strict]

Data Types

cell_list list
module_name string
cell_name string
block_name string
mim_module_name string

ARGUMENTS

cell_list

Specifies the list of cells that will be grouped into a new level of hierarchy. All of the cells must be instances in the same module. If more than one cell is specified, the list must be enclosed in braces ({}). Cell names cannot be specified with the following options: -hdl_block, -hdl_all_blocks, -hdl_bussed

-module_name *module_name*

Specifies the name to be given to the newly-created module. The specified name is taken as a suggestion; if it is not available, the tool generates a module name automatically and prints a warning message.

If *module_name* is not specified or if the specified name is not available, the tool automatically generates a name of the form "group_0", "group_1", etc.

-cell_name *cell_name*

Specifies the name to be given to the cell that instantiates the new hierarchy. The specified name is taken as a suggestion; if it is not available in the parent module, the tool generates a cell name automatically and prints a warning message.

If *cell_name* is not specified or if the specified name is not available, the tool automatically generates a name of the form "group_cell_0". Since a single counter is used for all tool-generated cell names, the number in the suffix might not start at 0 and might not be sequential across multiple invocations of the command.

-phys_wrapper

Specifies that a single hierarchical cell is being grouped for use with design planning, possibly as a step toward creating a new physical hierarchy. If this option is specified, the new module's ports and nets will have the same names as the corresponding ports and nets in the specified cell, rather than inheriting their names from the parent module's nets. It is an error to specify more than one cell, or to specify a non-hierarchical cell, when using this option.

While the wrapper cell inherits the port and net names from the inner cell, any constraints on those objects are not moved to the wrapper cell.

-hdl_block *block_name*

Groups all cells of a HDL block created in the RTL file. The -hdl_block option requires that you specify HDL block label name in the current instance. Nested HDL block labels are separated by a /(slash); for example, my_process/my_function. block_name must reference a single label name. Wildcards are supported for label names, but they do not match hierarchy characters. If wildcards match multiple labels, the first block with cells is used.

-hdl_all_blocks

Groups all HDL blocks into a level of hierarchy. If only -hdl_all_blocks option is used, it recursively groups all the HDL blocks in the current hierarchy. If used with -hdl_block, all HDL blocks from the block_name label down are recursively grouped.

-hdl_bussed

Places each group of bussed gates created from HDL into a level of hierarchy. If used with -hdl_block option, blocks specified by block_name are recursively grouped.

-enable_mim *mim_module_name*

Specifies the MIM module name for multi instance group_cells. The tool checks and tries to match the groupcells with the target MIM module. It even tries to match them by making possible connection changes. If they are not compatible, it errors out or continues with normal group_cells based on the switch '-mim_strict'.

-mim_strict

This switch directs the MIM group_cells flow to error out or disable in case of errors. If *-mim_strict* is present it errors out else gives warning and continues with the normal group_cells.

DESCRIPTION

This command groups any number of cells in the same parent module into a new module, and instantiates that module as a cell instance in the original parent module. The cells do not have to be in the current block or current instance, but all cells to be grouped have to be in the same parent module. A collection containing the newly-created cell is returned.

If *-phys_wrapper* is not specified, the new module's ports will have the same name as the nets to which they are connected in the parent module, and the connected nets inside the new module will have the same name as their ports. Cells and other nets within the new module retain their original names (but the hierarchical paths from the top module to the grouped nets and cells will be different).

The new module's port directions are determined automatically based on the driver and load pins connected both inside and outside the new hierarchy.

This command operates on a single instance of the parent module that contains the cells being grouped. If that module is multiply

instantiated, a separate, unique module will be created for this instance and the new hierarchy will appear only in that instance's module.

For -hdl_block, -hdl_all_blocks, and -hdl_bussed options, only HDL blocks within the current instance can be referenced. To reference HDL blocks inside submodules, use the current_instance command to change the current instance to the parent block containing the HDL block. References to recursive grouping in the ARGUMENTS section refer to nested labels within the current instance, not to logical design hierarchy.

EXAMPLES

The following example groups two cells in parent cell "mid1" into a new module "BOT" with cell instance "bot1":

```
prompt> group_cells -module_name BOT -cell_name bot1 {mid1/U1 mid1/U2}
{bot1}
prompt> get_cells mid1/bot1/*
{mid1/bot1/U1 mid1/bot1/U2}
```

The following example groups all cells named "decoder_*" into a new module, allowing the tool to generate the new module and cell names:

```
prompt> group_cells [get_cells decoder_*]
{group_cell_4}
prompt> current_instance group_cell_4
group_cell_4
prompt> get_cells
{group_cell_4/decoder_1 group_cell_4/decoder_0}
```

The following example groups all cells in all the HDL blocks into separate level of hierarchy (or modules);

```
prompt> current_instance bot1
prompt> group_cells -hdl_all_blocks
{group_cell_0 group_cell_1 ...}
```

The following example groups all cells in the HDL function named bar and the process named proc into a module:

```
prompt> current_instance bot1
prompt> group_cells -hdl_block proc/bar
{group_cell_0}
```

The following example groups all bussed gates under the process named proc into separate level of hierarchy (or modules):

```
prompt> current_instance bot1
prompt> group_cells -hdl_block proc -hdl_bussed
{group_cell_0 group_cell_1 ...}
```

The following example shows usage of group_cells with MIM and mim_strict;,

```
prompt> current_instance bot1
prompt> group_cells -cell_name mm1 -enable_mim hier1 -mim_strict
{group_cell_0 group_cell_1 ...}
prompt> group_cells -cell_name mm1 -enable_mim hier1 -mim_strict
{group_cell_3 group_cell_4 ...}
```

SEE ALSO

ungroup_cells(2)

group_path

Groups paths for cost function calculations and reporting.

SYNTAX

status **group_path**
-name *group_name*
[-default]
[-weight *weight_value*]
[-critical_range *range_value*]
[-from *from_list*]
[-rise_from *rise_from_list*]
[-fall_from *fall_from_list*]
[-through *through_list*]
[-rise_through *rise_through_list*]
[-fall_through *fall_through_list*]
[-to *to_list*]
[-rise_to *rise_to_list*]
[-fall_to *fall_to_list*]
[-priority *priority_level*]
[-comment *comment*]

Data Types

group_name string
weight_value float
range_value float
from_list list
rise_from_list list
fall_from_list list
through_list list
rise_through_list list
fall_through_list list
to_list list
rise_to_list list
fall_to_list list
priority_level list
comment string

ARGUMENTS

-name *group_name*

Specifies a name for the path group.

If a group with this name already exists, the paths are added to that group. If a group with this name does not exist, a new group is created.

The **-name** and **-default** options are mutually exclusive; you must specify one of these options.

-default

Removes the specified endpoints or paths and moves them to the default group.

The **-name** and **-default** options are mutually exclusive; you must specify one of these options.

-weight *weight_value*

Specifies a cost function weight for this group.

The *weight_value* argument must be a number between 0.0 and 100.0; the default is 1.0.

A weight of 0.0 eliminates the paths in this group from cost function calculations. Do not use very small values such as 0.0001. Smaller values can prevent the tool from implementing small improvements to the design.

If you use the **-weight** option when you add members to an existing group, the tool uses the new weight for the group.

-critical_range *range_value*

Specifies a margin of delay for this group during optimization.

The *range_value* argument must be positive or 0.0; the default is defined by the **set_critical_range** command, whose default is 0.0.

A range of 0.0 means that only the most critical paths (the ones with the worst violation) are optimized. If you specify a nonzero range, other near-critical violating paths (one per endpoint) within that amount of the worst path are also optimized if possible.

To optimize more than one critical path to an endpoint, use a separate **group_path** command for each distinct critical path to that endpoint. To optimize all violating paths, specify a range value that is larger than any expected path violation.

-from *from_list*

Specifies the timing path startpoints. A valid startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified.

If you specify a clock, all registers and primary inputs related to that clock are used as path startpoints. If you specify a cell, one path startpoint on that cell is affected.

You can use only one of the following options: **-from**, **-rise_from**, and **-fall_from**.

-rise_from *rise_from_list*

Specifies the timing path startpoints. This option is similar to the **-from** option, except that the path must rise from the specified objects.

If you specify a clock object, this option selects the startpoints launched by the rising edge of the clock at its source, taking into account any logical inversions along the clock path.

You can use only one of the following options: **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Specifies the timing path startpoints. This option is similar to the **-from** option, except that the path must fall from the specified objects.

If you specify a clock object, this option selects startpoints launched by the falling edge of the clock at its source, taking into account any logical inversions along the clock path.

You can use only one of the following options: **-from**, **-rise_from**, and **-fall_from**.

-through ***through_list***

Specifies the timing path throughpoints. A valid throughpoint is a port, pin, or leaf cell. If you specify more than one throughpoint, they must be enclosed in either quotation marks ("") or braces ({}).

The path grouping applies only to paths that pass through one of the throughpoints.

You can use multiple **-through** options in a single command. If you use multiple **-through** options, the path grouping applies to paths that pass through each set of throughpoints in the order in which they were specified. In other words, the path must first pass through a throughpoint specified in the first **-through** option, then through a throughpoint specified in the second **-through** option, and so on for every **-through** option.

If you the **-through** option with the **-from** or **-to** options, the path grouping applies only if all the conditions are satisfied.

-rise_through ***rise_through_list***

Specifies the timing path throughpoints. This option is similar to the **-through** option, but applies only to paths with a rising transition at the specified objects.

You can use multiple **-rise_through** options in a single command.

-fall_through ***fall_through_list***

Specifies the timing path throughpoints. This option is similar to the **-through** option, but applies only to paths with a falling transition at the specified objects.

You can use multiple **-fall_through** options in a single command.

-to ***to_list***

Specifies the timing path endpoints. A valid endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified.

If you specify a clock, all registers and primary outputs related to that clock are used as path endpoints. If you specify a cell, one path endpoint on that cell is affected.

You can use only one of the following options: **-to**, **-rise_to**, and **-fall_to**.

-rise_to ***rise_to_list***

Specifies the timing path endpoints. This option is similar to the **-to** option, but applies only to paths that rise at the endpoint.

If you specify a clock object, this option selects the endpoints captured by the rising edge of the clock at its source, taking into account any logical inversions along the clock path.

You can use only one of the following options: **-to**, **-rise_to**, and **-fall_to**.

-fall_to ***fall_to_list***

Specifies the timing path endpoints. This option is similar to the **-to** option, but applies only to paths that fall at the endpoint.

If you specify a clock object, this option selects the endpoints captured by the falling edge of the clock at its source, taking into account any logical inversions along the clock path.

You can use only one of the following options: **-to**, **-rise_to**, and **-fall_to**.

-priority ***priority_level***

Assigns a priority to the command to resolve conflicting path selections between different **group_path** commands. By default, different **group_path** commands follow ordinary rules of priority to resolve conflicting path selections. For example, a command

using **-from** and **-to** to select paths has priority over a conflicting command that selects a subset of those paths using **-through**, so the latter command is ignored for the paths that could be in either. By setting a higher priority in the command using the **-through** option, its path selection is honored, overriding the usual order of priority on paths in the intersection of the two groups. If unspecified, the priority is 0. You can set the priority for a **group_path** command to any integer 0 or greater.

-comment ***comment***

Specifies a comment string for the command.

The tool honors the annotation and preserves it with the SDC object so that the exact string is written out when the constraint is written out by the **write_sdc** or **write_script** command. The comment remains intact throughout the design flow.

DESCRIPTION

This command groups a set of paths or endpoints for cost function calculations in optimization and analysis. The delay cost function is the sum of all groups (weight * violation), where violation is the cost of the worst path in the path group. If no violation occurs within a group, the group cost is zero. The weight of the default group is 1.0.You can use path groups to optimize even though there might be larger violations in another group.

The **create_clock** command automatically creates a path group for a new clock with a weight of 1.0 and the same name as the clock name. By default, clock-gating check and asynchronous preset and clear check paths will be added to the clock group they relate to.

The **time.use_special_default_path_groups** application option may be used to alter this behavior.

In presence of special inbuild path group, the tool follows the following precedence rules for assigning group paths:

Order of precedence from highest to lowest is:

a) User defined group path

b) Special inbuilt group path (async and clock_gating only) **set_app_options -name time.use_special_default_path_groups -value true**

c) IO group path **set_app_options -name time.enable_io_path_groups -value true**

d) Default clock group

By default, the tool allows pulling paths from special built-in path groups like: **async_default**, **clock_gating_default** into user-defined path groups, which is different from PrimeTime. To match PrimeTime precedence rule for group path, use the following application option:

set_app_options -name time.special_path_group_precedence primetime

To remove a path group, use the **remove_path_groups** command. To report path group information for a design, use the **report_path_groups** command.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example groups all endpoints clocked by CLK1A or CLK1B into a new group named group1 with a weight of 2.0.

```
prompt> group_path -name group1 \
  -weight 2.0 -to {CLK1A CLK1B}
```

The following example adds OUT1 and ff34/D to the existing path group named ADDR.

```
prompt> group_path -name ADDR \
  -to {OUT1 ff34/D}
```

The following example removes OUT1 and CLK2 from existing groups and places them in the default group.

```
prompt> group_path -default \
  -to {OUT1 CLK2}
```

The following example groups all paths from inputs I1 and I2 to outputs O5 and O7 into a group named serious with a weight of 10.0.

```
prompt> group_path -name serious \
  -weight 10.0 -from {I1 I2} -to {O5 O7}
```

The following example groups the paths that begin at A1, pass through B1, then pass through C1, and end at D1 into a group named group2.

```
prompt> group_path -name group2 \
  -from A1 -through B1 -through C1 -to D1
```

The following example groups the paths that begin at A1, pass through either B1 or B2, then pass through either C1 or C2, and end at D1 into a group named group3.

```
prompt> group_path -name group3 \
  -from A1 -through {B1 B2} -through {C1 C2} -to D1
```

SEE ALSO

create_clock(2)
current_design(2)
remove_path_groups(2)
report_constraints(2)
report_timing(2)
reset_design(2)
set_input_delay(2)
set_output_delay(2)
time.enable_io_path_groups(3)
time.special_path_group_precedence(3)
time.use_special_default_path_groups(3)

gui_add_annotation

Adds an annotation to the layout window.

SYNTAX

status **gui_add_annotation**
[-window *window_name*]
[-group *group_type*]
[-type *shape_type*]
[-symbol_type *symbol_type*]
[-symbol_size *symbol_size*]
[-text *text*]
[-color *color*]
[-pattern *pattern*]
[-width *line_width*]
[-line_style *line_style*]
[-info_tip *info_tip*]
[-query_text *query_text*]
[-query_command *query_command*]
[-radius *radius*]
points

Data Types

window_name string
group_type string
shape_type string
symbol_type string
symbol_size string
text string
color string
pattern string
line_width string
line_style string
info_tip string
query_text string
query_command string
radius float
points list

ARGUMENTS

-window *window_name*

Specifies the instance name of the layout window. If window name is omitted, the annotation will apply to all layout windows.

-group *group_type*

Specifies the annotation group. If group name is omitted, the **global** name is used.

-type *shape_type*

Specifies the type of annotation. The supported values for this option are

rect
line
arrow
polygon
polyline
text
symbol
ruler
manhattan_ruler
circle
circular_ruler

The default is rect.

-symbol_type *symbol_type*

Specifies the type of symbol annotation. This option is only valid when annotation type is symbol. The supported values for this option are

square
x
triangle
diamond

The default is square.

-symbol_size *symbol_size*

Specifies the size of symbol annotation. An even symbol size will grow one automatically to find the appropriate center point. The default symbol size is 3.

-text *text*

Specifies the annotation text.

-color *color*

Specifies the annotation color. You can use a predefined color string, such as white, red, green, blue, yellow, and so forth, or a hexadecimal RGB value, such as #AA6633. The default annotation color is white. Annotations use colors from the qt color palette by default. Colors from other color palettes may be used by specifying a color name qualified with a palette name. For example, "red:highlight" specifies the red color from the highlight palette. Available palettes are qt, highlight, highcontrast, and style. Use gui_get_color_value command to report on color names and values available in a palette.

-pattern *pattern*

Specifies the annotation fill pattern. The default fill pattern is none. An empty value forces the tool to use the window default value, which is usually set to solid fill.

-width *line_width*

Specifies the annotation line width. The default line width is 1.

-line_style *line_style*

Specifies the annotation line style. The default line style is none. An empty value forces the tool to use the window default value, which is usually set to solid line.

-info_tip *info_tip*

Specifies an info tip for this annotation. When the user starts the query tool in the layout, and puts the mouse cursor over this annotation the specified text will be displayed as the infoTip.

If the text starts with a '=' character the text after it will be considered to be a tcl command which, when executed, will return the query text to be displayed. The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

%window the layout window name
%view the layout view name
%object a collection containing the annotation

-query_text *query_text*

This option is only valid if an info tip was specified. Use this to specify a more detailed string that is displayed in the query palette when the object is selected via the query tool. If query text is not specified then the query tool will just use the info tip string.

If the text starts with a '=' character the text after it will be considered to be a tcl command which, when executed, will return the query text to be displayed. The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

%window the layout window name
%view the layout view name
%object a collection containing the annotation

-query_command *query_command*

This option is only valid if an info tip was specified. Use this to specify a tcl command to be run when the user clicks on the annotation.

The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

%window the layout window name
%view the layout view name
%object a collection containing the annotation
%x the x position clicked in design coordinates
%y the y position clicked in design coordinates

-radius *radius*

Specifies the radius of circle annotation. This option is only valid when annotation type is circle.

points

Specifies the points for the specified annotation type. The points are specified by a tool command language (Tcl) list in which each list element is one of the following:

- the {x y} location of a point
- a collection containing a single object that has a visual representation in the layout. In other words, it is not a net, for example. The location of the point is the origin of the object.

If a collection is specified the list element can optionally include a position type, one of:

- center : the annotation is placed at the center of the largest rectangle of the object. The largest rectangle is the rectangle with the most area which is completely contained inside the object shape.
- bbox_center : the annotation is placed at the center of the bounding box of the object.
- bbox_ll : the annotation is placed at the lower left of the bounding box of the object.
- bbox_lr : the annotation is placed at the lower right of the bounding box of the object.
- bbox_ul : the annotation is placed at the upper left of the bounding box of the object.
- bbox_ur : the annotation is placed at the upper right of the bounding box of the object.

If a bounding box position type is specified, i.e. one of 'bbox_center', 'bbox_ll', 'bbox_lr', 'bbox_ul' or 'bbox_ur', then the list element can also optionally include an x and y fractional offset from this bounding box point. The x and y offset is a fraction of the width and height of the object's bounding box respectively. For example, for a 'bbox_center' position type, an x offset of -0.5 is the left edge and an x offset of 0.5 is the right edge. Similarly a y offset of -0.5 is the bottom edge and a y offset of 0.5 is the top edge.

For annotations referring to a single-object collection, if the object is moved, the tool updates the point automatically as needed.

DESCRIPTION

This command defines a new annotation and returns a collection containing that annotation if it is successful.

EXAMPLES

Create a green rectangle annotation.

```
prompt> gui_add_annotation -window Layout.1 \
  -type rect -color green {{200 200} {777 777}}
```

Create a light_green rectangle annotation using light_green from the highcontrast palette.

```
prompt> gui_add_annotation -window Layout.1 \
  -type rect -color light_green:highcontrast {{200 200} {777 777}}
```

Create a light_green x annotation of size 100 using light_green from the highcontrast palette.

```
prompt> gui_add_annotation -window Layout.1 \
  -type x -symbol_size 100 -color light_green:highcontrast [list {1835.0010 119.2950}]
```

Create a red polyline annotation in the group Test1.

```
prompt> gui_add_annotation -window Layout.2 \
  -group Test1 -type polyline -color red \
  -width 2 {{123 456} {789 12} {200 200}}
```

Create a line between two cells A and B.

```
prompt> gui_add_annotation -window Layout.1 \
  -type line [list [get_cells A] [get_cells B]]
```

Create a line between the center of largest rectangle of cell A and the center of the largest rectangle of cell B.

```
prompt> gui_add_annotation -window Layout.1 \
  -type line [list [list [get_cells A] center] [list [get_cells B] center]]
```

Create a line between the bottom left of cell A's bounding box and the middle of the top of cell B's bounding box.

```
prompt> gui_add_annotation -window Layout.1 -type line \
  [list [list [get_cells A] bbox_ll] [list [get_cells B] bbox_center 0.0 0.5]]
```

Create a blue rectangle which displays the annotation points using the annotation_query tcl procedure.

```
prompt> proc annotation_query { object window view } { \
 return [get_attribute $object points] \
 }

prompt> gui_add_annotation -window Layout.1 -type rect -color blue \
 -query_text {=annotation_query %object %window %view} -info_tip {annotation points} \
 {{110 110} {160 160}}
```

Create a circle annotation whose center is {100 100} and radius is 10.

```
prompt> gui_add_annotation -type circle -radius 10 {{100 100}}
```

Create a circle annotation whose center is {100 100} and boundary is at point {120 120}

```
prompt> gui_add_annotation -type circle {{100 100} {120 120}}
```

Create a circular ruler annotation with 3 circles (radius 20, 40, 100) at 100, 100.

```
prompt> gui_add_annotation -type circular_ruler {{100 100} {120 100} {140 100} {200 100}}
```

SEE ALSO

gui_remove_annotations(2)
gui_remove_all_annotations(2)
gui_get_color_value(2)

gui_add_hotkey_binding

Add additional hotkey binding to a menu item.

SYNTAX

string **gui_add_hotkey_binding**
-menu *MenuName*
-hot_key *HotKey*
[-replace_current_hotkey]
[-menu_root *MenuRoot*]

MenuName *String*
HotKey *String*
MenuRoot *String*

ARGUMENTS

-menu *MenuName*

MenuName specifies the menu item that the hotkey is associated with.

-hot_key *HotKey*

HotKey specifies the accelerator to be associated with the specified menu item.

-replace_current_hotkey

Specifies that the specified hotkey will replace the existing hotkey of the specified menu item.

-menu_root *MenuRoot*

MenuRoot specifies the menu tree that the menu item belongs to.

DESCRIPTION

This command adds or replace a hotkey binding for the specified menu item.

EXAMPLES

The following example creates a menu item and add an additional hotkey to it with this command. Note that the -menu option value in gui_add_hotkey_binding can have the ampersands stripped.

```
gui_add_menu -menu "&Test->Sub&Menu->Echo Hi" -tcl_cmd "echo hi" -hot_key "Ctrl+5"
gui_add_hotkey_binding -menu "Test->SubMenu->Echo Hi" -hot_key "Ctrl+Y"  #add additional hotkey to the menu item.
```

SEE ALSO

gui_add_menu(2)
gui_remove_toolbar(2)
gui_add_toolbar_item(2)
gui_remove_toolbar_item(2)
gui_show_toolbar(2)
gui_hide_toolbar(2)
gui_get_toolbar_names(2)

gui_add_missing_vias

Adds missing vias for the selected wires or nets.

SYNTAX

status **gui_add_missing_vias**
[-min_layer *min_layer_name*]
[-max_layer *max_layer_name*]
[-between_adjacent_layers_only]
[-quiet]
[*object_list*]

Data Types

min_layer_name string
max_layer_name string
object_list collection

ARGUMENTS

-min_layer *min_layer_name*

Specifies the name of the minimum bottom layer to consider when adding vias.

-max_layer *max_layer_name*

Specifies the name of the maximum upper layer to consider when adding vias.

-between_adjacent_layers_only

Specifies that vias should be added only between adjacent routing layers. No generation of via stacks is allowed. This option is equivalent to using of sequence of gui_add_missing_vias with adjacent layers specified in -min_layer and -max_layer options for all pairs of adjacent layers

-quiet

Suppress all messages.

object_list

Specifies the net name or net shape for which to create the via. If *object_list* is not specified, the command uses the global selection. If *object_list* is not specified and nothing is selected, the command issues an error message and does not modify the design.

DESCRIPTION

This command adds missing vias for unconnected wires in the GUI. Specify the wires for via insertion by selecting them in the GUI or by specifying the net names or net shapes to the command. For each selected wire, or wire on the selected net, the tool checks for overlaps with another wire, a pin shape or a port shape/terminal on a different layer on the same net. If an overlap is found and no via is present, the command ensures that vias exist so that these objects are connected.

The command considers the following:

Route Type

The route type of the new via is taken from the route type of the existing wire being connected. If there are two wires then the route type of one of the wires is used - which one is chosen is indeterminate.

Connection Point

The new via will be centered on the connection point between the two objects: wire-to-wire, wire-to-pin shape, or wire-to-port shape/terminal. The connection point is calculated by determining the overlap of the two shapes. The command snaps the center of the overlap using the layers and widths of the connecting objects.

Limitations

Vias inserted by this command on PG net wires are limited to only the selected wires and are not allowed for the whole net. This allows you to perform minor repairs and reduces the chance that you will accidently attempt to repair the whole power mesh. Repairing the entire power mesh would be prohibitively slow with this command and is better achieved by using separate power planning commands.

The vias created between PG net wires only use the simplified algorithms available in other route editing tools, such as the Stretch Wire tool and Create Via auto mode. If you need more control over via creation, use the **create_pg_vias** command or other tools available in the GUI.

EXAMPLES

The following example adds missing wires for all the wires on the net N1 between METAL and METAL3 layers.

```
prompt> change_selection [get_nets N1]
prompt> gui_add_missing_vias -min_layer METAL -max_layer METAL3
```

SEE ALSO

create_pg_vias(2)
get_edit_setting(2)
get_vias(2)
remove_vias(2)

gui_append_utable

Append data rows to an existing UserTable.

SYNTAX

string **gui_append_utable**
-name *Name*
-rows *TCL_Data_List*
-fill_file_column *Name*
-fill_file_suffix *Suffix*
-fill_file_dir *Directory*
-fill_label_column *Name*
-fill_label_value *Value*
-fill_date_column *Name*
-fill_tag_column *Name*
-fill_title_column \fiName

Name String
TCL_DataList TCL List of String List
Suffix File Suffix String
Directory Directory Path String
Value Value String

ARGUMENTS

-name *Name*

The name of an existing UserTable that the new data rows should be added to.

-rows *TCL_Data_List*

A TCL list of string lists that match the number of columns.

-fill_file_column *Name*

The column name that is filled with data from one of the fill options.

-fill_file_suffix *Name*

A file suffix/extension 'filter' used to fill file entries for the column defined by the -fill_column argument.

-fill_file_dir *Directory*

The directory path to search for files matching the file suffix option.

-fill_label_column *Name*

The column name that is filled with a label for each entry added.

-fill_label_value *Value*

The label value used to fill the -fill_label_column argument as each entry is added.

-fill_date_column *Name*

The column name used to fill in the date of the file that was added.

-fill_tag_column *Name*

The column name used to fill in the metadata tag of the file if it exists.

-fill_title_column *Name*

The column name used to fill in the metadata title of the file if it exists.

DESCRIPTION

This command takes a TCL list of lists and tries to append new rows of data to an existing UserTable given by name.

Row columns are filled left to right, first to last, in a given list of lists.
Extra data values in a row list are ignored and if the row list is too short on values it will fill the extra columns with empty fields.

If the **-rows** argument is just a single list of values, it is assumed to be only one row of values that will be added. Again extra row values are ignored and missing row values are filled with empty fields.

Alternatively one can generate rows of data that include columns for file names, file dates, labels and if available in the file the tag and title meta data for all files found according to the specified file suffix and directory.
For more information on Meta Data please see the command *gui_set_utable_meta*.

EXAMPLES

The following example appends two rows of static data given as a TCL list of lists to the given UserTable that has three columns of data.

```
shell> gui_append_utable -name my_table {
...     { /top/ModA  0.01  1000 }
...     { /top/ModB  0.03  5000 }
...   }
```

The following example first creates a TCL list of lists and then appends it to the given UserTable.

```
shell> set rows {}
```

```
shell> ... loop through collection and set vars and append to list...
shell> lappend rows [list $cellname $delay $area]
shell> ...
shell> gui_append_utable -name MyTable -rows $rows
```

SEE ALSO

gui_close_utable(2)
gui_change_selection_utable(2)
gui_create_utable(2)
gui_export_utable(2)
gui_get_utable(2)
gui_import_utable(2)
gui_set_utable_meta(2)
gui_open_utable(2)
gui_show_utable(2)
gui_write_utable(2)

gui_bin

Organizes a collection of objects into bins.

SYNTAX

string **gui_bin**
-clct *Clct*
[-attr *Attribute* | -cmd *Command*]
[-lower_bound *Lower_Bound*]
[-lower_bound_strict]
[-upper_bound *Upper_Bound*]
[-upper_bound_strict]
[-boundary *Boundary*]
[-num_bins *Number_of_Bins* | -bin_range *Range_per_Bin*]
[-underflow]
[-overflow]
[-nice_level *Nice_Level*]
[-small_is_good]
[-exact_binning]
[-ignore_values *Ignore_List*]
[-bar_brush *Brush_Pattern*]
[-create_slct_buses]
[-filter_cmd *Filter_Command*]
[-numBin *numBins*]
[-return_values]
[-slct_targets *Selection_Targets*]
[-slct_targets_operation *Selection_Operation*]
[-value_list *Element_Value_List*]

Data Types

Clct collection
Attribute string
Command string
Lower_Bound float
Upper_Bound float
Boundary float
Number_of_Bins integer
Range_per_Bin float
Nice_Level integer
Ignore_List list
Brush_Pattern string
Filter_Command string
numBins integer
Selection_Target list
Selection_Operation string
Element_Value_List list

ARGUMENTS

-clct *Clct*

Specifies the collection of objects that the tool partitions into bins.

-attr *Attribute*

Specifies the attribute that the tool uses to get a floating point value for each object in the collection. The tool places the objects in bins based on these attribute values.

The **-attr** and **-cmd** options are mutually exclusive.

-cmd *Command*

Specifies the tool command language (Tcl) command that the tool uses to get a floating point value for each object in the collection. The tool places the objects in bins based on these values.

The tool replaces all occurrences of %clct in the command with a collection that contains, as its only element, the object used to compute the floating point value for the object. If the tool cannot store the object in a collection, it uses the string "Error" instead.

The tool replaces all occurrences of %string in the command with the name of the object. If the tool cannot compute the name for an object, it uses the string "Error" instead.

The **-cmd** and **-attr** options are mutually exclusive.

-lower_bound *Lower_Bound*

Defines the lower bound for the range of floating point values that the tool uses to place the objects in bins.

If you specify the **-underflow** option, the tool places all the objects with a value that is less than *Lower_Bound* in a separate underflow bin. If you do not specify the -underflow option, the tool discards these objects.

-lower_bound_strict

Forces the tool to use the **-lower_bound** option value as the lower bound of the bin with the smallest values, not including the underflow bin.

If you do not specify the -lower_bound_strict option, the tool sets the lower bound of the bin with the smallest values, not including the underflow bin, to the smallest value that is greater than or equal to the value specified by the **-lower_bound** option.

-upper_bound *Upper_Bound*

Defines the upper bound for the range of floating point values that the tool uses to place the objects in the bins.

If you specify the **-overflow** option, the tool places all the objects with a value that is greater than *upper_Bound* in a separate overflow bin. If you do not specify the **-overflow** option, the tool discards these objects.

-upper_bound_strict

Forces the tool to use the **-upper_bound** option value as the upper bound of the bin with the largest values, not including the overflow bin.

If you do not specify the **-upper_bound_strict** option, the tool sets the upper bound of the bin with the largest values, not including the overflow bin, to the largest value that is less than or equal to the value specified by the **-upper_bound** option.

-boundary *Boundary*

Specifies the boundary value for one of the bins, not including the underflow and overflow bins. This value must fall within the range of values covered by the bins.

If the relevant value distribution has a specific value where the quality of the values change, such as 0 for slack, you can use this option to make sure that no bin contains values on both sides of the boundary value.

-num_bins ***Number_of_Bins***

Specifies the number of bins that this command generates, not counting underflow and overflow bins. The default value is 8.

The **-num_bins** and **-bin_range** options are mutually exclusive. If you specify both of these options, the tool ignores the **-bin_range** option. If you do not specify either of these options, the default value is 8.

-bin_range ***Range_per_Bin***

Specifies the size of the value range represented by each bin.

The **-bin_range** and **-num_bins** options are mutually exclusive. If you specify both of these options, the tool ignores the **-bin_range** option. If you do not specify either of these options, the default value is 8.

-underflow

Collects all the objects with values less than the lower bound value specified by the **-lower_bound** option, and places them in a separate underflow bin. Otherwise, these objects are discarded.

-overflow

Collects all the objects with values greater than the upper bound value specified by the **-upper_bound** option, and places them in a separate overflow bin. Otherwise, these objects are discarded.

-nice_level ***Nice_Level***

Determines how round the boundaries of the bins are. The default value is 10. For rounder boundaries, specify a higher value. If *Nice_Level* is -1, the tool does not attempt to make the boundaries round.

-small_is_good

Sorts the values in every bin in order of decreasing values. By default, the tool sorts the objects in order of increasing values.

In addition, bins with values that are less than the value specified by the **-boundary** option are colored green, and bins with values that are greater than this value are colored red. This is opposite from the default color markings.

-exact_binning

Sets exact boundaries on the left border of the furthest left bin and the right border of the furthest right bin. By default, the tool can adjust these limits to create bins with equal widths.

-ignore_values ***Ignore_List***

Discards the objects with values specified in *ignore_List*.

-bar_brush ***Brush_Pattern***

Specifies the brush pattern for painting bars in a histogram based on the result of the command. The *brush_Pattern* can be one of the following values:

NoBrush SolidPattern Dense1Pattern
Dense2Pattern Dense3Pattern Dense4Pattern
Dense5Pattern Dense6Pattern Dense7Pattern
HorPattern VerPattern CrossPattern
BDiagPattern FDiagPattern DiagCrossPattern

-create_slct_buses

Creates selection buses to return the objects instead of collections.

-filter_cmd *Filter_Command*

Specifies a Tcl command that the **gui_bin** command uses to determine whether a collection in a bin should be filtered out.

The *filter_Command* format is

cmd %clct %attr %value

where *cmd* is a Tcl command or procedure, *%clct* is a collection, *%attr* is the name of the attribute used to create the bins, and *%value* is the maximum value for *%clct*. If the command returns a nonzero value, **%clct** is not placed in a bin.

-numBin *numBins*

This option is obsolete. Do not use it. Use the **-num_bins** option instead.

-return_values

Causes the **gui_bin** command to return the values of elements.

-slct_targets *Selection_Target*

Specifies a list of selection containers to store the objects.

-slct_targets_operation *Selection_Operation*

Specifies an operation applied to the selection containers.

-value_list *Element_Value_List*

Specifies values for the elements in a list. The **gui_bin** command uses these values to create the bins.

DESCRIPTION

The **gui_bin** command creates a list of bins from a collection of objects that you specify with the **-clct** option. The command determines a floating point value for each of the objects by using either an attribute that you specify with the **-attr** option or a Tcl command that you specify with the **-cmd** option.

Each bin is represented by a list that contains the left boundary, the right boundary, the number of objects in the bin, a collection containing the objects in the bin, and the value green or red.

The widths of the bins are equal by default, which means that in some cases the furthest left and furthest right bins might extend the specified boundaries in order to get nice values for the boundaries. If you specify the **-exact_binning** option, the left border of the furthest left bin and the right border of the furthest right bin are set exactly to the specified boundaries.

The collections are sorted with ascending floating point values by default, and a bin is marked green only if the objects it contains have values greater than or equal to the value specified by the **-boundary** option. If you specify the **-small_is_good** option, the values in the collections are sorted with descending floating point values, and a bin is marked red only if the objects it contains have values greater than or equal to the value specified by the **-boundary** option.

The bins cover the complete range of floating point values by default. You can define the covered range by using the -lower_bound, -lower_bound_strict, -upper_bound, and **-upper_bound_strict** options. In this case, you can collect objects with floating point values that are too small or too big and place them into separate underflow and overflow bins by specifying the **-underflow** and **-overflow** options.

You can specify a list of objects with floating point values that can be ignored by using the **-ignore_values** option.

You can specify a bin range size that results in a very large number of bins by using the **-bin_range** option. However, this results in a histogram that is difficult to read and that might take a long time to compute and draw.

The preference variable **hist_max_num_bins** in the Histogram category allows you to set the maximum number of histogram bins. The default value is 100. If a specified range size value causes the number of bins to exceed the value set for this variable, the tool issues an error message and does not create a histogram. The error message includes information about the data range to aid you in specifying a better -bin_range value. This variable is used only in conjunction with the **-bin_range** option and is not used with the **-num_bins** option.

EXAMPLES

The following example creates a list of bins containing the 100 worst paths according to their slack distribution:

```
prompt> set binInfo [gui_bin -clct [get_timing_paths \
  -nworst 100 -max_paths 100] -cmd "get_attribute %clct slack"]
prompt> set bins  [lindex $binInfo 0]
```

SEE ALSO

gui_change_charts_model

Change data in model used in chart's plots

SYNTAX

status **gui_change_charts_model**
-model *model_id*
[-column *string*]
{ -add | -delete | -modify | -calc | -query }
[-header *string*]
[-type *string*]
[-expr *string*]

Data Types

model_id int

ARGUMENTS

-model *model_id*

Model to change (previously created by **gui_create_charts_model** command).

-column *string*

The column name or number to delete, modify, calculate or query.

-add

Add column to model. The *expr* option specifies the expression to use to calculate the values in the new column.

-delete

Delete column from model. Use the *column* option to specify the column to delete.

Note: the original columns of the model cannot be deleted, only newly added columns can be removed after they are created.

-modify

Modifies the values in an existing column. Use the *column* option to specify the column to modify and the *expr* option to specify the expression to use to calculate the new values in the column.

Note: the original columns of the model cannot be modified, only newly added columns can be modified are created.

-calc

Calculate values from model. Use the *column* option to specify the default column for the values and the *expr* option to specify the

expression to use to calculate the values.

The calculated values are returned by the command. The model is not modified.

-query

Query values from model. Use the *column* option to specify the default column for the values and the *expr* option to specify the expression to use to query the values.

The expression should return true or false and the returned values are those where the expression evaluates to true.

The queried values are returned by the command. The model is not modified.

-header *string*

Specifies the name of the header when adding or modifying a column.

-type *string*

Specify the type of the new column for *add*, *modify* options.

-expr *string*

Specifies the expression to use when adding, modifying, calculating or querying values of a column of the model.

The expression can use tcl variable names that match the column name (as long as the column name is a legal tcl variable name). A few extra variables are also allowed:

- column : current column
- row : current row
- pi : value of pi (3.141592653...)
- NaN : Special 'not a number' real value that can be used to indicate absence of data.

A number of extra functions are allowed as well as the normal tcl expression functions:

- column : get column value for row
- row : get row value for column
- cell : get cell value for row and column
- setColumn : set column value for row
- setRow : set row value for column
- setCell : set cell value for row and column
- header : get header value for column
- setHeader : set header value for column
- type : get type for column
- setType : set type for column
- map : map row number to range
- bucket : get bucket for column value
- norm : normalize column value

- rand : get random number
- rnorm : get normaized random number
- color : get color from name
- remap : map column value to range
- timeval : get time value from column with time format

Some special character sequences in the expression are replaced with model values before the expression is evaluated.

- @<n> : value for specified column number (<n>) for current row.
- @c : column number.
- @r : row number.
- @nc : number of columns.
- @nr : number of rows.
- @v : value for current row and column.
- @{<name>} : value for specified column name (<name>) for current row.
- #{<name>} : column number for specified column name (<name>)

Normally the '@' symbols return data in the value format but this can be forced to be a string by using '@#' instead of '@'.

DESCRIPTION

Change data in existing model to add, delete or modify columns for use in charts.

EXAMPLES

The following example adds a new column to the model where the value is the sum of the first two columns.

```
shell> gui_change_charts_model -model $model -add -expr {column(0) + column(1)} -header Sum
```

SEE ALSO

gui_create_charts_plot(2)
gui_define_charts_proc(2)

gui_change_error_highlight

Manipulates error objects highlight sets.

SYNTAX

string **gui_change_error_highlight**
-add | -remove
[-color *color_id*]
[-selected | -all | -all_visible]
[-type *error_type_list*]
[-layer *layer_strings*]
[-include_net *nets* | -exclude_net *nets*]

color_name A color name
error_type_list A Tcl list of error type names
layer_strings A Tcl list of layer strings
nets A Tcl list of net names or a collection of nets

ARGUMENTS

-add | -remove

Required mutually exclusive option which specifies that the errors are added or removed from the error highlight set. If -add, the specified errors become drawn in the given highlight color. If -remove, the specified errors become drawn in the default violation color.

-color <*color_id*>

This option specifies the color for the operation. If a color is given with the -color option, it must be one of the supported color names: blue, green, light_blue, light_green, light_orange, light_purple, light_red, orange, purple, red, yellow. If -color option is provided with the -add option, the specified color will be used to color the specified errors in the given color. If -color option is provided with the -remove option, errors in the specified color highlight set will be removed from the highlight set and will become drawn in the default violation color. When given with the -remove option, this option is mutually exclusive with -selected, -all, -all_visible, -type, -layer, -include_net, and exclude net options.

-selected

This option is mutually exclusive with -all, -all_visible, -type, -layer, -include_net and -exclude_net. If this option is present, currently selected errors in the current tab error list become highlighted in the specified color.

-all

This option is only meaningful with the -remove and is mutually exclusive with -color, -all_visible, and -selected. When given with the -remove option, removes highlight colors from all errors in all open error views.

-all_visible

This option is mutually exclusive with -selected, -type, -layer, -include_net and -exclude_net. If this option is given with the -add option, currently visible errors in the current tab error list become highlighted in the specified color. If this option is given with the -remove option, all error highlight colors are removed from currently visible errors in the current tab error list. They become drawn in the default violation color.

-type <*error_type_list*>

This option is mutually exclusive with -selected and -all_visible. It may be combined with -layer, and -include_net or -exclude_net. Errors that are of one of the given types become highlighted in the specified color. If combined with layer and/or net specifications, errors must match all given specifications to be selected for highlighting.

-layer <*layer_string_list*>

This option is mutually exclusive with -selected and -all_visible. It may be combined with -type, and -include_net or -exclude_net. Errors that are of one of the given layers become highlighted in the specified color. If combined with error type and/or net specifications, errors must match all given specifications to be selected for highlighting.

-include_net <*nets*>

This option is mutually exclusive with -selected, -all_visible and with -exclude_net. It may be combined with -type and -layer. Errors that are associated with one of the given nets become highlighted in the specified color. NULL net is specified with an empty net name: "". If combined with error type and/or layer specifications, errors must match all given specifications to be selected for highlighting.

-exclude_net <*nets*>

This option is mutually exclusive with -selected, -all_visible and with -include_net. It may be combined with -type and -layer. Errors that are not associated with one of the given nets become highlighted in the specified color. NULL net is specified with an empty net name: "". If combined with error type and/or layer specifications, errors must match all given specifications to be selected for highlighting.

DESCRIPTION

The command specifies the color to use in drawing the specified errors. Errors can be drawn in a highlight color by adding them to the error highlight color set. Errors can be removed from the error highlight color set and revert to being drawn in the default violation color. The command behavior is determined by combinations of the options as follows:

Adding errors to a highlight set:

If -color is given with the -add option, the specified errors are added to the error highlight color set and become draw in the specified color. The specified color becomes the current error highlight color.

If -color is omitted with the -add option, the specified errors are added to the current error highlight color set and become draw in the current error highlight color.

If -selected is given with the -add option, selected errors are added to the error highlight color set and become draw in the specified color.

If -all_visible is given with the -add option, all visible errors in the current tab error list are added to the error highlight color set and become draw in the specified color.

If -type, -layer -include_net or -exclude_net is given with the -add option, visible errors in the current tab error list matching the given criteria are added to the error highlight color set and become drawn in the specified color.

Removing errors from a highlight set:

If -color is given with the -remove option, all errors in the specified error highlight color set are removed and become drawn in the

default violation color.

If -selected is given with the -remove option, selected errors are removed from respective error highlight color sets and become drawn in the default violation color.

If -all is given with the -remove option, all errors are removed from respective error highlight color sets and become drawn in the default violation color.

If -all_visible is given with the -remove option, all visible errors in the current tab error list are removed from respective error highlight color sets and become drawn in the default violation color.

If -type, -layer -include_net or -exclude_net is given with the -remove option, visible errors in the current tab error list matching the given criteria are removed from the respective error highlight color set and become drawn in the default violation color.

EXAMPLES

The following example puts the currently selected errors in the blue error highlight set:

gui_change_error_highlight -add -color blue -selected

The following example puts "Short" type errors associated with a net named "C0_9_" in the red error highlight set:

gui_change_error_highlight -add -color red -type {Short} -include_net {C0_9_}

The following example puts "Floating Port" type errors not associated with NULL net in the green error highlight set. Please note the use of parentheses as delimiters. Since the expected argument for -type and -exclude_net options is a list of strings, multi-word strings or empty string signifying NULL net must be delimited:

gui_change_error_highlight -add -color green -type {{Floating Port}} -exclude_net {{}}

The following example adds all visible errors in the current tab error list to the light_red highlight set:

gui_change_error_highlight -add -color light_red -all_visible

The following example removes all errors from all error highlight sets:

gui_change_error_highlight -remove -all

The following example removes errors in the red color highlight set from the highlight set:

gui_change_error_highlight -remove -color red

The following example removes all visible errors in the current tab error list from highlight sets:

gui_change_error_highlight -remove -all_visible

SEE ALSO

gui_set_selected_errors(2)

gui_change_highlight

Manipulate the set of globally highlighted objects.

SYNTAX

string **gui_change_highlight**

[-add | -remove | -toggle]
[-color *color_id* | -all_colors]
[-collection *clct*]

string *color_id*
collection *clct*

ARGUMENTS

-add

Use -add to highlight a collection of objects in a color specified with color. If -color is not specified, the current color is used. You cannot use -all_colors with -add. You must specify a collection with -collection. If -remove or -toggle are not specified, then -add is implied.

-remove

Use -remove to remove highlighting from objects. Use -collection to remove highlighting from specific objects. Use -color to remove highlighting from objects of a specific color. Use -all_colors to remove all highlighting. If you specify a collection, you cannot specify colors.

-toggle

Use -toggle to toggle the highlight state of a collection of objects. You must use -collection with -toggle, and you cannot use -color or -all_colors. Toggling an object that is highlighted will cause it to no longer be highlighted. Toggling an object that is not highlighted with cause it to be highlighted with the current color. If no operation is specified, then -add is assumed.

-color *color_id*

Specifies the color to be used for the highlight operation. This is mutually exclusive with -all_colors. The allowed colors are blue, light_blue, yellow, purple, light_purple, orange, light_orange, red, light_red, green, and light_green. If neither -color nor -all_colors is specified, then the current highlight color is assumed. You cannot use -color with -toggle.

-all_colors

This option is only valid with -remove. Use this option to clear all highlight colors.

DESCRIPTION

The gui_change_highlight command is used to operate on the set of highlighted objects. Objects can be added or removed from the set, or their presence can be toggled as described above. The set has a current color that can be modified with gui_set_highlight_options(2). The current color is used as needed when no color is specified.

The tool provides eleven built in highlight colors with the names yellow, orange, red, green, blue, purple, light_orange, light_red, light_green, light_blue, and light_purple.

The gui_change_highlight command uses colors from the highlight color palette.

EXAMPLES

Highlight in green all cells with names matching foo*.

```
shell> gui_change_highlight -color green -collection [get_cells foo* -hier -all]
```

Remove highlights from all objects in the collection returned by the get_cells command.

```
shell> gui_change_highlight -remove -collection [get_cells foo* -hier -all]
```

Clear all objects with blue highlights.

```
shell> gui_change_highlight -remove -color blue
```

Clear objects with the current highlight color.

```
shell> gui_change_highlight -remove
```

Clear all highlights.

```
shell> gui_change_highlight -remove -all_colors
```

Toggle the highlight state of all objects in a collection returned by the get_cells command.

```
shell> gui_change_highlight -toggle -collection [get_cells foo*]
```

SEE ALSO

gui_get_highlight(2)
gui_get_highlight_options(2)
gui_set_highlight_options(2)
gui_get_color_value(2)

gui_change_layer

Sets the layer of pins, ports, or net shapes.

SYNTAX

collection **gui_change_layer**

-object *object*
-layer *layer*

Data Types

object collection
layer string

ARGUMENTS

-object *object*

Specifies the collection of objects for which to change the layer.

-layer *layer*

Specifies the name of the layer to use.

DESCRIPTION

The **gui_change_layer** command is used to change the layer of block pins, ports/terminals, and net shapes. Changing the layer of net shapes and terminals creates new shape objects and deletes the existing shapes. The **gui_change_layer** command updates the selection to contain the new objects. Use the **set_object_layer** command instead to avoid updating the selection.

EXAMPLES

The following example changes the layer of the clk_enable block pin.

prompt> **gui_change_layer -object [get_pins TOP/LEVEL_1/clk_enable] -layer METAL2**

SEE ALSO

get_layers(2)
set_object_layer(2)

gui_change_schematic

Performs abstraction on the current or specified schematic view.

SYNTAX

string **gui_change_schematic**
  -type *abstraction_name*
  -operation *operation_name*
  -clct *objects*
  [-window *window_name*]

Data Types

abstraction_name	string
operation_name	string
objects	collection
window_name	string

ARGUMENTS

-type *abstraction_name*

Name of abstraction type. Currently supported types: "Hierarchy".

-operation *operation_name*

Name of operation to be applied for the specified abstraction type. Currently supported operations: "Expand", "Collapse".

-clct *objects*

The collection of objects in the target schematic to which the specified abstraction operation will be applied.

-window *window_name*

Optionally specifies the target schematic to which the abstraction is applied. If not specified, the current application schematic will be the target.

DESCRIPTION

The command applies the specified abstraction operation to the specified objects within the target schematic view.

EXAMPLES

The following example collapses currently selected modules within the current schematic view.

prompt> **gui_change_schematic -type Hierarchy -operation Collapse -clct [get_current_selection]**

gui_change_selection_utable

Change Current Selection by adding objects named in the given table.

SYNTAX

string **gui_change_selection_utable**
-name *Table_Name*
[-obj_col *Col_Name*]
[-filter *Filter*]

Table_Name String
Col_Name String
Filter Table Filter String

ARGUMENTS

-name *Table_Name*

The name of the user table to use for finding object names to add to the current selection.

-obj_col *Col_Name*

The column name containing object names with a data type set. By default it uses the full_name and object_class type or named columns.

-filter *Filter*

This argument allows you to filter the rows in the given table to only rows that match the filter. The filter expression is the same expression allowed in the GUI filter field of the UserTable GUI view.

DESCRIPTION

This command takes a table name and if a filter is given, reduces the rows to those that match the filter and then tries to add to the current selection any object names found in those table rows. The object names are searched for in the column given by the *-obj_col* command argument. By default it uses the full_name and object_class type or named columns to determine object names to add to the current selection.

EXAMPLES

The following example adds all object with the object names found in the column 'pins' that match the filter '*ALU*'.

```
shell> gui_change_selection_utable -name my_table -obj_col pins -filter {pins =~ *ALU*}
```

SEE ALSO

gui_append_utable(2)
gui_close_utable(2)
gui_create_utable(2)
gui_export_utable(2)
gui_import_utable(2)
gui_open_utable(2)
gui_show_utable(2)
gui_write_utable(2)

gui_change_via_def

Sets the via definition for the specified vias.

SYNTAX

string **gui_change_via_def**
-via *via*
-via_def *via_def*

Data Types

via collection
via_def string

ARGUMENTS

-via *via*

Specifies the collection of vias for which to change the via definition.

-via_def *via_def*

Specifies the name of the new via definition.

DESCRIPTION

The **gui_change_via_def** command is used to change the via definition of vias. By changing the via definition, new vias with different via definitions might be created. The **gui_change_via_def** command updates the selection to contain the new vias. Use the **set_via_def** command instead to avoid updating selection.

EXAMPLES

The following example changes the via definitions on a via named VIA_S_92587 to VIA34.

prompt> **gui_change_via_def -via [get_vias VIA_S_92587] -via_def VIA34**

SEE ALSO

create_via_def(2)
get_via_defs(2)
get_vias(2)
report_via_defs(2)
set_via_def(2)

gui_change_via_size

Sets the attributes for the number of rows and columns in a via collection.

SYNTAX

status **gui_change_via_size**
-via *via*
[-rows *rows*]
[-columns *columns*]

Data Types

via collection
rows integer
columns integer

ARGUMENTS

-via *via*

Specifies the collection of vias for which to change the array size.

-rows *rows*

Specifies the number of rows in the via array.

-columns *columns*

Specifies the number of columns in the via array.

DESCRIPTION

The **gui_change_via_size** command changes the number of rows and columns of vias. By changing the via size, new vias with a different via definitions might be created. The **gui_change_via_size** command updates the selection to contain the new vias. Use the **set_via_def** command instead to avoid updating the selection.

EXAMPLES

The following example changes the array size on a via named VIA_S_92587 to two rows and two columns.

```
prompt> gui_change_via_size -via [get_vias VIA_S_92587] -rows 2 -columns 2
```

SEE ALSO

get_vias(2)
set_via_def(2)

gui_check_drc_errors

Checks for DRC errors on specified objects by using the same DRC mechanism that the interactive route editing tools use.

SYNTAX

int **gui_check_drc_errors**
[-of_objects *objects*]
[-limit_area {*{llx lly} {urx ury}*}]
[-layers *layer_list*]
[-honor_ndr true | false]
[-rules *rule_types*]
[-filter_same_net_spacing true | false]
[-check_fill true | false]
[-max_error_limit *error_limit*]
[-max_shape_limit *shape_limit*]
[-max_processing_time *time_limit*]
[-show]

Data Types

objects collection
llx float
lly float
urx float
ury float
layer_list collection
rule_types list
error_limit integer
shape_limit integer
time_limit integer

ARGUMENTS

-of_objects *objects*

Specifies a collection of net shape and via objects on which the tool performs the DRC checks. By default the DRC check is performed on the selection.

-limit_area {*{llx lly} {urx ury}*}

Specifies a rectangular area in which the tool performs the DRC checks. The tool checks only those shapes of the specified objects that are enclosed within the area or that intersect the bounding box that defines the area.

-layers *layer_list*

Specifies the layers which to check. By default, the tool checks objects on all layers.

-honor_ndr true | **false**

Specifies whether the tool checks the supported nondefault routing rules. If this is not specified, the return value from **get_edrc_setting -honor_ndr** is used as default.

-rules ***rule_types***

Specifies the types of DRC rules that the tool checks. The *rule_types* list can contain one or more of the following DRC error types:

Adjacent Min Edge Length
All
DPT Odd Cycle
DPT Pre-color
Enclosed Via Spacing
End-of-Line Spacing
General Via Spacing
Metal Span Spacing
Metal Width
Minimum Edge
Minimum Length and Area
Open
RDL Acute Angle
RDL Right Angle
Short
Spacing
Via Density
Via Enclosure

All can be used as the list of all other rules except *Open*. If you do not specify this option, the tool uses those relevant option values returned by the **get_edrc_setting** command.

-filter_same_net_spacing true | **false**

Specifies whether the tool ignores spacing violations between objects on the same net. If this is not specified, the return value from **get_edrc_setting -filter_same_net_spacing** is used as default.

-check_fill true | **false**

Specifies whether the tool to check shapes inside fill cells. If this is not specified, the return value from **get_edrc_setting -check_fill** is used as default.

-max_error_limit ***error_limit***

Overrides the maximum number of DRC errors that the tool reports.

A zero or a negative value allows an unlimited number of errors.

If this option is not specified, the return value from **get_edrc_setting -max_error_limit** is used as default.

If the *error_limit* or default limit is reached, the tool issues a warning message in the terminal window, the console log view, and output log file, and then stops writing out further messages.

-max_shape_limit ***shape_limit***

Overrides the maximum number of shapes that the tool can check.

A zero value allows an unlimited number of shapes.

If this option is not specified, the return value from **get_edrc_setting -max_shape_limit** is used as default.

If the number of objects to be checked exceeds *shape_limit*, the command exits without checking. A warning message is issued in the terminal window and the console log view.

-max_processing_time *time_limit*

Specifies the maximum processing time in seconds that the tool can check. If this is not specified, the return value from **get_edrc_setting -max_processing_time** is used as default. The DRC checking terminates when the time limit is reached.

-show

Open the error browser to show violations.

DESCRIPTION

This command performs design rule checks on the specified objects and returns the number of violations found.

The command uses the same DRC mechanism that the route editing tools use during interactive editing operations. The tool writes the violations in the editor DRC error view and loads it into the error browser when error browser opens.

The editor DRC error view is associated with the current top-level design. This error view remains open after error checking finishes to ensure the validity of the returned collection of objects with violations. The tool closes the editor DRC error view automatically if you close the design or end the GUI session.

The editor DRC error view associated with the top-design is named *current_top_design_CEL_name*_edrc.err by default. You can retrieve the name of the editor DRC error view, including the version, as an attribute of the top-level design. The attribute name is **editor_error_view_name**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example checks the selected objects for DRC errors, writes the violations into the editor DRC view, and loads the view into the error browser.

```
prompt> gui_check_drc_errors -show
```

The following example checks the net shapes inside the region *{{0 0} { 50 50} }* on METAL1 layer for *Minimum Edge* rule.

```
prompt> gui_check_drc_errors -limit_area {{0 0} { 50 50} }\
  -rules {"Minimum Edge"} -layers METAL1 -show
```

SEE ALSO

get_drc_errors(2)

get_edrc_setting(2)
gui_error_browser(2)
remove_drc_errors(2)
set_edrc_setting(2)

gui_clear_error_data_filter

Remove the filter from error data.

SYNTAX

string **gui_clear_error_data_filter**

DESCRIPTION

Remove the filter and restore visibility of errors. Errors may remain hidden based on null net association or fixed status state if the "Hide Null-Net Errors" option or the "Hide Fixed Errors" option has been set.

EXAMPLES

The following example removes the error data filter

```
gui_clear_error_data_filter
```

SEE ALSO

gui_error_browser(2)
gui_set_error_data_filter(2)
gui_set_error_browser_option(2)

gui_clear_selected_errors

Clears selection in the Error Browser.

SYNTAX

string **gui_clear_selected_errors**

ARGUMENTS

DESCRIPTION

Clears selected errors in the Error Browser.

EXAMPLES

The following example clears selected errors:

```
gui_clear_selected_errors
```

SEE ALSO

gui_error_browser(2)
gui_set_current_errors(2)
gui_set_selected_errors(2)

gui_close_error_data

Removes an error data file from the error browser.

SYNTAX

status **gui_close_error_data**
[*error_data*]

Data Types

error_data collection

ARGUMENTS

error_data

A collection of physical DRC error data objects to remove from the error browser. A collection of the error data or the error data name can be used to remove it from the error browser.

DESCRIPTION

The **gui_close_error_data** command removes the given error data from the GUI error browser. If no error data is given, then the command removes all error data from the error browser and hides the error browser.

When a command opens an error data, it increments the open_count for an error data. Incrementing the open_count needs to be balanced by the same number of decrementing the open_count before the error data is closed and removed from memory.

The **gui_open_error_data** command opens an error data if necessary. It does not increment the open_count if the error data is already open and in memory. If the **gui_open_error_data** command increments the open_count, then a subsequent call to **gui_close_error_data** for the same error data decrements the open_count. Otherwise, a subsequent call to **gui_close_error_data** simply removes the error_data from the error browser without decrementing the open_count.

EXAMPLES

The following example removes the physical DRC error data file that is named "my_design_dppinassgn.err" from the error browser.

```
prompt> gui_close_error_data "my_design_dppinassgn.err"
1
```

The following example removes all physical DRC error data file from the error browser and hides the error browser.

```
prompt> gui_close_error_data
1
```

SEE ALSO

gui_open_error_data(2)
gui_get_error_data(2)
close_drc_error_data(2)
open_drc_error_data(2)
save_drc_error_data(2)
get_drc_error_data(2)
create_drc_error_data(2)
remove_drc_error_data(2)
collections(2)

gui_close_utable

Close and free the given list of UserTables in memory.

SYNTAX

string **gui_close_utable**
-names *Table_Names*

Table_Names String List

ARGUMENTS

-names *Table_Names*

The list of table names to close and free in memory.

DESCRIPTION

This command closes and frees all the table names given by the user. If the table was in read only mode (streaming), then the connection to the table file is closed.

EXAMPLES

The following example closes a list of user tables.

shell> **gui_close_utable -names {my_table1 my_table2}**

SEE ALSO

gui_append_utable(2)
gui_change_selection_utable(2)
gui_create_utable(2)

gui_export_utable(2)
gui_import_utable(2)
gui_open_utable(2)
gui_show_utable(2)
gui_write_utable(2)

gui_close_window

Close the specified window

SYNTAX

status **gui_close_window**
{ -window *window_id* | -type *window_type* | -all }
[-exact_type_match]

Data Types

window_id string
window_type string

ARGUMENTS

-window *window_id*

Specifies the window name id.

Any matching window of this name will be closed.

This option precludes the *-type* and *-all* options.

-type *window_type*

Specifies the window type id. Any matching window of this type will be closed.

If the *-exact_type_match* options is not specified then types derived from this window type will also be closed. Derived types are created with the **gui_create_window_type** command.

This option precludes the *-window* and *-all* options.

-all

Specifies that we want to close all the windows.

This option precludes the *-window* and *-type* options.

-exact_type_match

Specifies that types derived from the specified type are excluded when the *-type* option is specified.

Note: this option should only be used with the *-type* option.

DESCRIPTION

This command closes the specified window(s).

Windows can be specified by name, type or as all windows.

EXAMPLES

The following examples will create a toplevel window and a child layout view window, then close the command layout window.

```
shell> set top [gui_create_window -type TopLevel]
shell> set x [gui_create_window -type Layout -parent $top]
shell> gui_close_window -window $x
```

SEE ALSO

gui_create_window(2)
gui_exist_window(2)
gui_show_window(2)
gui_get_window_types(2)
gui_get_window_ids(2)
gui_set_active_window(2)
gui_get_current_window(2)

gui_connect_charts_signal

Connect to charts signal

SYNTAX

status **gui_connect_charts_signal**
{ -view *view_name* | -plot *plot_name* }
-from *signal_name*
-to *tcl_proc*

Data Types

view_name string
plot_name string
signal_name string
tcl_proc string

ARGUMENTS

-view *view_name*

Charts view to connect to.

-plot *plot_name*

Plot to connect to.

-from *signal_name*

Signal name to connect to.

Supported names are:

objIdPressed (id proc) plot object clicked on with mouse

annotationIdPressed (id proc) plot or view annotation clicked on with mouse

chartObjsAdded (non-id proc) objects added to chart

-to *tcl_proc*

Tcl procedure to call when the signal is emitted.

For plot an id procedure is called with three arguments (view, plot and object_id) and a non-id procedure is called with two arguments (view, chart).

For view an id procedure is called with two arguments (view, object_id) and a non-id procedure is called with one arguments

(view).

DESCRIPTION

Connects signals emitted from charts events to tcl callbacks.

EXAMPLES

The following example calls chartObjPressed proc when object in plot pressed.

shell> gui_connect_charts_signal -plot $plot -from objIdPressed -to chartObjPressed

SEE ALSO

gui_create_charts_plot(2)

gui_copy_charts_model

Copy existing charts model to new model

SYNTAX

status **gui_copy_charts_model**
-model *model_id*

Data Types

model_id int

ARGUMENTS

-model *model_id*

Charts model to copy.

DESCRIPTION

Copy data in existing charts model to a new model.

EXAMPLES

The following example copies an existing model to a new one.

shell> set model2 [gui_copy_charts_model -model $model1]

SEE ALSO

gui_create_charts_model(2)
gui_change_charts_model(2)

gui_create_attrgroup

Creates a group of attributes for an object type.

SYNTAX

status **gui_create_attrgroup**
-class *design_object*
-name *name*
[-attr_list {{*attr_1*}...{*attr_n*}}]

Data Types

design_object	string
name	string
attr_1	string
attr_n	string

ARGUMENTS

-class *design_object*

Specifies the type of design object.

-name *name*

Specifies the name of the attribute group to be created for the specified object type.

-attr_list {{*attr_1*}...{*attr_n*}}

Lists and orders the attributes to be included in the group.

DESCRIPTION

The **gui_create_attrgroup** command creates a new attribute group for a specific type of design object. The command adds the attribute group to the display in GUI tools, such as the Properties dialog box, List views, layout view InfoTips, and other graphic windows.

An attribute can be assigned to more than one attribute group at the same time.

The order of the attributes in an attribute group is important and is set by the **-attr_list** option when you create the group. Use the **gui_update_attrgroup** command if you need to change or reorder the attribute list for an attribute group.

EXAMPLES

```
prompt> gui_create_attrgroup -class Cell -name "Hierarchy" \
  -attr_list { "Name" "Hierarchical" "Owning Cell"
  "isPhysConstraint" "Hard Macro" "Num Children" "Num Nets"
  "Num Pins" "Utilization" "Area" "Aspect Ratio" "Min Aspect"
  "Max Aspect" "Min Area" "Max Area" "Area / 1M" "Hier Req Area / 1M"
  "Reference" "Logical Name" "FullName" }

prompt> gui_create_attrgroup -class Cell -name "Edit" \
  -attr_list { "Name" "isPhysConstraint" "Orientation"
  "Location" "Bounding Box" "Fixed Location" "Is Resizable"
  "Hard Macro" "Placed" "Is IO" "Off Litho Grid" "FullName" }

prompt> gui_create_attrgroup -class Pin -name "Timing" \
  -attr_list { "Name" "Is Clock" "Slew"  "slack_max" }

prompt> gui_create_attrgroup -class Net -name "Timing" \
  -attr_list { "Name" "Lumped C" "Total Lumped C"
  "Lumped Resistance" }
```

SEE ALSO

gui_delete_attrgroup(2)
gui_list_attrgroups(2)
gui_update_attrgroup(2)

gui_create_block_diagram

Creates a block diagram view for the hierarchy cells specified by cell names or a collection. If no cell name or collection is specified, current selection is used. If there is no selection, the top-level design is used.

SYNTAX

string **gui_create_block_diagram**
[*collection*]
[-cells *cellNames*]
[-expanded]

Data Types

collection collection
cellNames list

ARGUMENTS

collection

Specifies a collection of hierarchy cells; the collection is used to initialize the block diagram view. If there is only one hierarchy cell in the collection, it is used as the root of the block diagram with all the hierarchy blocks it contains as the contents. If there are multiple hierarchy cells in the collection, a common hierarchy parent is determined and used as the root for block diagram and its contents are expanded to make the specified blocks visible.

This option is mutually exclusive with **-cells** option. Specify either a collection or a list of cell names but not both. If neither is specified, the current selection is used. In there is no selection, the top-level design is used.

-cells *cellNames*

Specifies a Tcl list of hierarchy cell names; the list is used to initialize the block diagram view. If only one hierarchy cell is specified, it is used as the root of the block diagram with all its child hierarchy blocks as the contents. If multiple hierarchy cells are specified, a common hierarchy parent is determined and used as the root for block diagram and its contents are expanded to make the specified blocks visible.

This option is mutually exclusive with the *collection* argument. Specify either a collection or a list of cell names but not both. If neither is specified, the current selection is used. In there is no selection, the top-level design is used.

-expanded

Display only buses in the block diagram without trying to bundle the buses together. By default, buses are bundled to create a more abstract block diagram view.

DESCRIPTION

The command generates a block diagram view of the currently selected design and hierarchy cells.

EXAMPLES

Each of the following examples generate a block diagram of hierarchical cell named RAM_1. The first command selects cell RAM_1. The second and third commands generate a block diagram of RAM_1 where its buses are bundled (not expanded). The last command generates a block diagram with buses the of RAM_1 unbundled.

```
prompt> change_selection [find cell RAM_1]
prompt> gui_create_block_diagram -cells {RAM_1}
prompt> gui_create_block_diagram [get_selection]
prompt> gui_create_block_diagram [get_selection] -expanded
```

SEE ALSO

get_selection(2)
change_selection(2)

gui_create_category_nodes

Creates one or more offspring category nodes under one or more parent category nodes by applying one or more category rules to the parent category nodes.

SYNTAX

gui_create_category_nodes
[-tree *category_tree*]
-parent_categories *category_nodes_list*
-rule_names *rule_names_list*

Data Types

category_tree	string or category_tree collection
category_nodes_list	list
rule_names_list	list

ARGUMENTS

-tree *category_tree*

The category tree in which the category node operation will happen; must be a category tree name or a category tree collection of size one; the most recently used category tree is used if the *-tree* option is omitted.

-parent_categories *category_nodes_list*

List of hierarchical (parent category node) name patterns and collections of parent category nodes under which one or more offspring category nodes will be created. A parent category node name is a unique hierarchical name of a category node in a Tcl list format.

-rule_names *rule_names_list*

List of one or more names of the category rules which will be applied incrementally to the parent category nodes to create new offspring category nodes.

DESCRIPTION

Creates one or more offspring category nodes under one or more parent category nodes by incrementally applying one or more category rules to the parent category nodes. Applying a single filter rule produces one new new offspring category node as well as an uncategorized offspring category labeled UNCAT (if UNCAT does not already exist under the parent category).

Applying a single dynamic category rule generates one or more new offspring category nodes. The command returns "1" if

successful; otherwise, an empty string is returned.

You should run this command only if at least one category tree is available. The command acts on the category tree specified.

EXAMPLES

This example shows the creation of one or more offspring category nodes by applying a single category rule to a single parent category node:

```
shell> gui_create_category_nodes \
? -parent_categories {{ALL **clock_gating_default**}} \
? -rule_names {{Start Point}} \
? -tree Tree1
1
```

This example is similar to the previous example; however, the optional *-tree* option is not specified. Therefore, the current category tree is used.

```
shell> gui_create_category_nodes \
? -parent_categories {{ALL **clock_gating_default**}} \
? -rule_names {{Start Point}}
1
```

This example shows the creation of one or more offspring category nodes by applying two category rules (in order) to a single parent category node:

```
shell> gui_create_category_nodes \
? -parent_categories {{ALL **clock_gating_default**}} \
? -rule_names {{Start Point} {Slack LT 0}}
1
```

This example shows the creation of offspring category nodes by applying a single category rule to multiple parent category nodes:

```
shell> gui_create_category_nodes \
? -parent_categories {{ALL {StartBlock: ALU}} {ALL {StartBlock: LAT}}} \
? -rule_names {EndBlock}
1
```

SEE ALSO

gui_remove_category_nodes(2)
gui_get_category_nodes(2)

gui_create_category_rule

Creates a category rule for automatic generation of one or more object categories.

SYNTAX

string **gui_create_category_rule**
[-name *rule_name*]
[-filter *filter_spec*]
-category *category_spec*
[-builtin]

Data Types

rule_name string
filter_spec string
category_spec string

ARGUMENTS

-name *rule_name*

Specifies the name of the category rule to be created. If this option is omitted, a unique new rule name is automatically generated.

-filter *filter_spec*

Specifies the filter. The basic form of a filter conditional expression is a series of relations joined together with && (and), || (or), and ! (not) operators. Parentheses are used to override precedence. The basic relation is either a Boolean attribute or a comparison of one non-Boolean attribute with another or with a constant value using relational operators.

For example, a filter expression can have any of the following forms:

- **-filter {endpoint_clock_is_propagated}** (Boolean attributes)
- **-filter {(slack < 0)}** (Comparison with a literal number)
- **-filter {(path_group.full_name == "inputs")}** (Comparison with a literal string)
- **-filter {(startpoint_clock.full_name == endpoint_clock.full_name)}** (Comparison of attributes)

Some object attributes have values that are object collections of size one. For those attributes, "dot notation" can be used to access attributes of the object in the collection. The following relational operators are supported:

- **==** (Equal)
- **!=** (Not equal)
- **=~** (Matches pattern)

- **!~** (Does not match pattern)
- **&~** (Contains all elements)
- |~ (Contains some elements)
- **>** (Greater than)
- **<** (Less than)
- **>=** (Greater than or equal to)
- **<=** (Less than or equal to)

The matching operators are used to match wildcard regular expressions. The containment operators are used to compare space-separated set or list attributes.

-category *category_spec*

Specifies the category. The category specification can be fixed literal text, or it can include an optional object attribute name enclosed in angle brackets; for example, **-category <path_type>**. Additional literal text is optionally allowed to the left and to the right of the angle brackets; for example **-category {Path Type: <path_type>}**. Some object attributes have values that are object collections of size one. For those attributes, "dot notation" can be used to access attributes of the object in the collection; for example, **-category <startpoint_clock.full_name>**.

When a category rule is "executed" later in the GUI, a separate category (bin of objects) is generated for each unique value of the attribute specified by the **-category** option. (The GUI "category-rule-execution engine" examines attribute values for an input set of objects.)

-builtin

Indicates that the rule being created belongs to the built-in group of category rules. If this option is omitted, the rule being created does not belong to the built-in group.

The built-in group of category rules is a group of category rules that are created before any non-built-in rules. This group of rules includes both built-in rules created by the tool (at tool startup time) and built-in rules that you create before creating any non-built-in rules.

DESCRIPTION

This command creates a category rule. If rule creation is successful, the command returns the name of the newly-created category rule; otherwise, it returns an empty string. After a category rule has been created, it can be used later (at "category-rule execution time" in the GUI) with other rules to generate one or more object categories.

EXAMPLES

This example creates a simple category rule:

```
prompt> gui_create_category_rule -name pathgroups \
    -category <path_group.full_name>
pathgroups
```

This example creates a simple filtering rule:

```
prompt> gui_create_category_rule -name {Negative Slack} \
    -category {Failing Paths} -filter {slack < 0}
Negative Slack
```

This example creates a regular expression filtering rule:

```
prompt> gui_create_category_rule -name {ATPG Sessions} \
    -category {ATPG Sessions} -filter {session_name =~ "atpg*"}
Negative Slack
```

This example creates a set or list containment filtering rule:

```
prompt> gui_create_category_rule -name {Through CPU and CTRL} \
    -category {Through CPU and CTRL} -filter { blocks &~ "CPU CTRL"}
Negative Slack
```

SEE ALSO

gui_list_category_rules(2)
gui_remove_category_rules(2)

gui_create_category_tree

Creates a new category tree and returns it in a collection of size one.

SYNTAX

category_tree_collection **gui_create_category_tree**
[-name *tree_name*]
-collections *collections_list*

Data Types

category_tree_collection collection (of one category tree object)
tree_name string
collections_list list (of collections)

ARGUMENTS

-name *tree_name*

Name of the category tree to be created. If this option is omitted, then a unique new category tree name will be automatically generated.

-collections *collections_list*

A list of collections of objects to be categorized. Each collection in the list should be homogeneous and each collection should be of the same object class, such as the timing path object class.

DESCRIPTION

Creates a new category tree and returns it as a collection of size one if category tree creation is successful (otherwise an empty string is returned). The new category tree holds onto the object collections passed to the command via -collections for future categorization.

A category tree is a collectible object of object class category_tree, and supports querying these attributes (which are defined only after gui_start) via get_attribute:

- name - The name of the category tree
- is_current - Is true if the category tree is the current category tree; is false otherwise.
  One and only one of the existing category trees is considered to be current at any time.

- object_class - This attribute always has the value "category_tree" for a category tree object.

EXAMPLES

This example show the creation of a category tree, and querying an attribute of the newly created category tree:

```
shell> set path_clct [get_timing_paths ...]
_sel18
shell> set path_clct2 [get_timing_paths ...]
_sel19
shell> set cat_tree \
?   [gui_create_category_tree -name tree0 -collections [list $path_clct $path_clct2]]
_sel20
shell> sizeof_collection $cat_tree
1
shell> get_attribute $cat_tree name
tree0
```

SEE ALSO

gui_get_category_trees(2)
gui_remove_category_trees(2)
get_attribute(2)

gui_create_charts_arrow_annotation

Create arrow annotation in view or plot

SYNTAX

arrow_id **gui_create_charts_arrow_annotation**
{ -view *view_name* | -plot *plot_name* }
[-id *string*]
[-tip *string*]
-start *position*
-end *position*
[-angle *angle*]
[-back_angle *angle*]
[-fhead *sbool*]
[-thead *sbool*]
[-properties *name_value_list*]

Data Types

view_name string
plot_name string
position string
angle float
name_value_list string
arrow_id annotation_id

ARGUMENTS

-view *view_name*

Charts View to add annotation to.

-plot *plot_name*

Plot to add annotation to.

-id *string*

Optional identifier string for annotation.

-tip *string*

Optional tip string for annotation.

-start *position*

Start point of the arrow line.

-end ***position***

End point of the arrow line.

-angle ***angle***

Angle of arrow head.

-back_angle ***angle***

Back angle of arrow head.

-fhead ***sbool***

Is arrow drawn at start of line.

-thead ***sbool***

Is arrow drawn at end of line.

-properties ***name_value_list***

Specifies a list of name value pairs for the properties to set on the created annotation.

RETURNS

arrow_id

Returns id of created annotation.

DESCRIPTION

Add an arrow annotation to plot or view.

The arrow annotation is a line with optional arrow heads draw at each end.

The line and arrow heads can be customized.

EXAMPLES

The following example creates an arrow annotation on a plot from (0 0) to (100 100) with arrows at both ends.

shell> gui_create_charts_arrow_annotation -plot $plot -start {0 0} -end {100 100} -fhead 1 -thead 1

SEE ALSO

gui_create_charts_plot(2)
gui_create_charts_ellipse_annotation(2)
gui_create_charts_point_annotation(2)
gui_create_charts_polygon_annotation(2)
gui_create_charts_polyline_annotation(2)
gui_create_charts_rectangle_annotation(2)
gui_create_charts_text_annotation(2)
gui_remove_charts_annotation(2)
gui_set_charts_property(2)
gui_get_charts_property(2)

gui_create_charts_correlation_model

Create correlation model from existing charts model

SYNTAX

status **gui_create_charts_correlation_model**
-model *model_id*

Data Types

model_id int

ARGUMENTS

-model *model_id*

Model to create correlation model for.

DESCRIPTION

Create correlation model from existing model.

The input model column values are compared to generate a symmetric correlation matrix of values between -1 and 1 where closely correlated column values have high values (positive or negative) and uncorrelated column values have low numbers.

The command returns the model index of the newly created model.

EXAMPLES

The following example creates a correlation of an existing model.

shell> gui_create_charts_correlation_model -model $model

SEE ALSO

gui_create_charts_model(2)

gui_create_charts_ellipse_annotation

Create ellipse annotation in view or plot

SYNTAX

ellipse_id **gui_create_charts_ellipse_annotation**
{ -view *view_name* | -plot *plot_name* }
[-id *string*]
[-tip *string*]
-center *position*
-rx *length*
-ry *length*
[-properties *name_value_list*]

Data Types

view_name string
plot_name string
position string
length float
name_value_list string
ellipse_id annotation_id

ARGUMENTS

-view *view_name*

Charts View to add annotation to.

-plot *plot_name*

Plot to add annotation to.

-id *string*

Optional identifier string for annotation.

-tip *string*

Optional tip string for annotation.

-center *position*

Center of the ellipse.

-rx *length*

X Radius of the ellipse.

-ry ***length***

Y Radius of the ellipse.

-properties ***name_value_list***

Specifies a list of name value pairs for the properties to set on the created annotation.

RETURNS

ellipse_id

Returns id of created annotation.

DESCRIPTION

Add an ellipse annotation to plot or view.

The background and border of the ellipse can be customized.

EXAMPLES

The following example creates an ellipse annotation on a plot centered at (0 0) with a radius of 60 in x direction and 40 in y direction.

shell> gui_create_charts_ellipse_annotation -plot $plot -center {0 0} -rx 60 -ry 40

SEE ALSO

gui_create_charts_plot(2)
gui_create_charts_arrow_annotation(2)
gui_create_charts_point_annotation(2)
gui_create_charts_polygon_annotation(2)
gui_create_charts_polyline_annotation(2)
gui_create_charts_rectangle_annotation(2)
gui_create_charts_text_annotation(2)
gui_remove_charts_annotation(2)
gui_set_charts_property(2)
gui_get_charts_property(2)

gui_create_charts_model

Create a model from data for use in charts

SYNTAX

model_id **gui_create_charts_model**
{ -csv *filename* | -clct *collection* |
-tcl *tcl_list* }
[{ -comment_header | -first_line_header }]
[-first_column_header]
[-transpose]
[-columns *column_names*]
[-column_type *column_type*]
[-name *string*]

Data Types

filename string
collection string
tcl_list string
column_names string
column_type string

ARGUMENTS

-csv *filename*

Create model from contents of specified csv filename.

A CSV file has comma separated fields.

The *comment_header*, *first_line_header* and *first_column_header* options can be used to determine which (if any) lines are used for horizontal and vertical headers.

-clct *collection*

Create model from contents of a collection. Each row consists of a list of attribute values from the collection object.

The *columns* option can be used to specified which attributes are used to populate the table from the collection. If the *columns* option is not specified then the attributes from the ALL attribute group are used.

-tcl *tcl_list*

Create model from the values in a tcl variable.

The tcl variable must be a list of lists. The data is a list of rows where each row is a list of column values.

The *transpose* option can be used to transpose the value array so lists of columns can be specified where each column is a list of row values.

-comment_header

Specifies that the first comment (line starting with a #) contains the names of the horizontal column headers.

Only used by the *csv* option.

-first_line_header

Specifies that the first line contains the names of the horizontal column headers.

Used by the *csv* and *tcl* options.

-first_column_header

Specifies that the first column (first field in each row) contains the names of the vertical row headers.

If the data also has a horizontal header then the first field of the horizontal header is ignored.

Used by the *csv* and *tcl* options.

-transpose

Specifies that the tcl value array from *tcl* option is transposed so lists of columns can be specified where each column is a list of row values.

-columns *column_names*

Specifies a list of names for the attributes to query from the collection to generate the columns of each row.

Used by the *clct* option.

-column_type *column_type*

Specifies the type and optional formatting of the values in the columns.

If the type of the column is not specified then the application will try to automatically determine it depending on whether the values can be all converted to real or integer values. If not the default string type is used.

The specified value is an ordered list of column type data. Each column type data is a list where the first element specifies the type and any extra elements specify extra name values to customize the type. By default, the type is associated with the column number matching the index of the item but a specific column can be specified by including the column name or number with the type value in the first element.

Syntax:

- column_type_data : { <column_type> [<name_values>]}
- column_type : { [<column>] <type> }
- column : column_name|column_index
- type : integer|real|string|...
- name_values : <name_value> <name_value> ...
- name_value : { <name> <value> }

The supported name values are dependant on the column type.

Column types can be queried using:

gui_get_charts_data -name column_types

The column types named values can be queried using:

gui_get_charts_data -name column_type.names -data $type

-name ***string***

Specifies the name of the model.

This can be used to annotate the model so the user can more easily identify what data it contains in the gui.

If the *csv* option is used then the filename is the default name.

RETURNS

model_id

The id of the created model.

This can be used as input to the **gui_create_charts_plot** command and other commands to do further processing on the model.

DESCRIPTION

This command creates a persistant model and loads data into it so it can be used as the data source for one or more plots.

The command returns a model id which can be used as input to the **gui_create_charts_plot** and other charts commands.

EXAMPLES

The following example creates a model from the csv file "test.csv" using the first line for the column headers.

shell> gui_create_charts_model -csv "test.csv" -first_line_header

The following example creates a model from a collection of all shapes with columns for the name and layer attributes.

shell> gui_create_charts_model -clct [get_nets] -columns [list name layer]

SEE ALSO

gui_create_charts_plot(2)

gui_create_charts_plot

Create a plot of the type using the supplied data

SYNTAX

plot_name **gui_create_charts_plot**
-view *view_name*
{ -clct *collection* | -model *model_id* }
-type *plot_type*
[-columns *column_names*]
[-title *string*]
[-properties *name_value_list*]
[-xmin *float*]
[-ymin *float*]
[-xmax *float*]
[-ymax *float*]

Data Types

view_name string
collection string
model_id int
plot_type string
column_names string
name_value_list string

ARGUMENTS

-view *view_name*

Parent view for plot.

-clct *collection*

Collection to use for input data. Each row consists of a list of attribute values from the collection object.

The *columns* option is used to specified which attributes are read from the collection to populate the table values.

Note: the resultant model is a snapshot of the specified collection's attribute values and will not updates as the collection is changed.

-model *model_id*

The model to use for input data.

-type *plot_type*

The type of plot to display:

The plot types can be queried using the command **gui_get_charts_data -name types**

-columns *column_names*

Specifies the columns to use for each plot column parameter.

The columns string is a command separated list of name value pairs of the form "<parameter name>=<column name>".

The parameter names are plot type dependent.

When the *model* option is specified the column names are column indices (0 based) or column header names.

When the *clct* option is specified the column names are attributes names.

-title *string*

Specifies the title for the plot.

-properties *name_value_list*

Specifies a list of name value pairs for the properties to set on the created plot.

-xmin *float*

User specified minimum x value (to override the auto calculated range).

-ymin *float*

User specified minimum y value (to override the auto calculated range).

-xmax *float*

User specified maximum x value (to override the auto calculated range).

-ymax *float*

User specified maximum y value (to override the auto calculated range).

RETURNS

plot_name

The id of the created plot.

DESCRIPTION

Create plot in a view using values in data model.

The view and model data must be created before the plot can be added.

EXAMPLES

The following example creates a distribution plot from the specified model using values from the first column.

```
shell> set window [gui_get_current_window -mru]
shell> set view [gui_create_window -parent $window -type Charts]
shell> gui_create_charts_plot -view $view -model $model -type distribution -columns "value=0"
```

SEE ALSO

gui_create_window(2)
gui_create_charts_model(2)

gui_create_charts_point_annotation

Create point annotation in view or plot

SYNTAX

point_id **gui_create_charts_point_annotation**
{ -view *view_name* | -plot *plot_name* }
[-id *string*]
[-tip *string*]
-position *position*
[-size *length*]
[-type *symbol_type*]
[-properties *name_value_list*]

Data Types

view_name string
plot_name string
position string
length float
symbol_type string
name_value_list string
point_id annotation_id

ARGUMENTS

-view *view_name*

Charts View to add annotation to.

-plot *plot_name*

Plot to add annotation to.

-id *string*

Optional identifier string for annotation.

-tip *string*

Optional tip string for annotation.

-position *position*

Point position.

-size *length*

Point size.

-type *symbol_type*

Point symbol type.

One of: dot, cross, plus, y, triangle, itriangle, box, diamond, star5, star6, circle, pentagon, ipentagon, hline, vline.

-properties *name_value_list*

Specifies a list of name value pairs for the properties to set on the created annotation.

RETURNS

point_id

Returns id of created point.

DESCRIPTION

Add a point annotation to plot or view.

The background and border of the point can be customized.

EXAMPLES

The following example creates a point annotation on a plot with a circle symbol at (0 0).

shell> gui_create_charts_point_annotation -plot $plot -position {0 0} -type circle

SEE ALSO

gui_create_charts_plot(2)
gui_create_charts_arrow_annotation(2)
gui_create_charts_ellipse_annotation(2)
gui_create_charts_polygon_annotation(2)
gui_create_charts_polyline_annotation(2)
gui_create_charts_rectangle_annotation(2)
gui_create_charts_text_annotation(2)
gui_remove_charts_annotation(2)
gui_set_charts_property(2)
gui_get_charts_property(2)

gui_create_charts_polygon_annotation

Create polygon annotation in view or plot

SYNTAX

poly_id **gui_create_charts_polygon_annotation**
{ -view *view_name* | -plot *plot_name* }
[-id *string*]
[-tip *string*]
-points *position_list*
[-properties *name_value_list*]

Data Types

view_name string
plot_name string
position_list string
name_value_list string
poly_id annotation_id

ARGUMENTS

-view *view_name*

Charts View to add annotation to.

-plot *plot_name*

Plot to add annotation to.

-id *string*

Optional identifier string for annotation.

-tip *string*

Optional tip string for annotation.

-points *position_list*

List of polygon points.

-properties *name_value_list*

Specifies a list of name value pairs for the properties to set on the created annotation.

RETURNS

poly_id

Returns id of created polygon.

DESCRIPTION

Add a polygon annotation to plot or view.

The background and border of the polygon can be customized.

EXAMPLES

The following example creates a polygon annotation on a plot with a red background with 50% alpha.

```
shell> gui_create_charts_polygon_annotation -plot $plot \
  -points "{0 0} {10 10} {20 40} {30 20} {40 10} {50 60} {60 40}" \
  -background 1 -fill_color red -fill_alpha 0.5
```

SEE ALSO

gui_create_charts_plot(2)
gui_create_charts_arrow_annotation(2)
gui_create_charts_ellipse_annotation(2)
gui_create_charts_point_annotation(2)
gui_create_charts_polyline_annotation(2)
gui_create_charts_rectangle_annotation(2)
gui_create_charts_text_annotation(2)
gui_remove_charts_annotation(2)
gui_set_charts_property(2)
gui_get_charts_property(2)

gui_create_charts_polyline_annotation

Create polyline annotation in view or plot

SYNTAX

poly_id **gui_create_charts_polyline_annotation**
{ -view *view_name* | -plot *plot_name* }
[-id *string*]
[-tip *string*]
-points *position_list*
[-properties *name_value_list*]

Data Types

view_name string
plot_name string
position_list string
name_value_list string
poly_id annotation_id

ARGUMENTS

-view *view_name*

Charts View to add annotation to.

-plot *plot_name*

Plot to add annotation to.

-id *string*

Optional identifier string for annotation.

-tip *string*

Optional tip string for annotation.

-points *position_list*

List of polygon points.

-properties *name_value_list*

Specifies a list of name value pairs for the properties to set on the created annotation.

RETURNS

poly_id

Returns id of created poly line.

DESCRIPTION

Add a multi-point line annotation to plot or view.

The background and border of the line can be customized.

EXAMPLES

The following example creates a polyline annotation on a plot with a red border with 50% alpha.

```
shell> gui_create_charts_polygon_annotation -plot $plot \
  -points "{0 0} {10 10} {20 40} {30 20} {40 10} {50 60} {60 40}" \
  -border 1 -stroke_color red -stroke_alpha 0.5
```

SEE ALSO

gui_create_charts_plot(2)
gui_create_charts_arrow_annotation(2)
gui_create_charts_ellipse_annotation(2)
gui_create_charts_point_annotation(2)
gui_create_charts_polygon_annotation(2)
gui_create_charts_rectangle_annotation(2)
gui_create_charts_text_annotation(2)
gui_remove_charts_annotation(2)
gui_set_charts_property(2)
gui_get_charts_property(2)

gui_create_charts_rectangle_annotation

Create rectangle annotation in view or plot

SYNTAX

rect_id **gui_create_charts_rectangle_annotation**
{ -view *view_name* | -plot *plot_name* }
[-id *string*]
[-tip *string*]
[-rectangle *string*]
[-properties *name_value_list*]

Data Types

view_name string
plot_name string
name_value_list string
rect_id annotation_id

ARGUMENTS

-view *view_name*

Charts View to add annotation to.

-plot *plot_name*

Plot to add annotation to.

-id *string*

Optional identifier string for annotation.

-tip *string*

Optional tip string for annotation.

-rectangle *string*

Rectangle bounding box

-properties *name_value_list*

Specifies a list of name value pairs for the properties to set on the created annotation.

RETURNS

rect_id

Returns id of created rectangle.

DESCRIPTION

Add a rectangle annotation to plot or view.

The background and border of the rectangle can be customized.

EXAMPLES

The following example creates a rectangle annotation on a plot from (0 0) to (100 100).

```
shell> gui_create_charts_rectangle_annotation -plot $plot -start {0 0} -end {100 100}
```

SEE ALSO

gui_create_charts_plot(2)
gui_create_charts_arrow_annotation(2)
gui_create_charts_ellipse_annotation(2)
gui_create_charts_point_annotation(2)
gui_create_charts_polygon_annotation(2)
gui_create_charts_polyline_annotation(2)
gui_create_charts_text_annotation(2)
gui_remove_charts_annotation(2)
gui_set_charts_property(2)
gui_get_charts_property(2)

gui_create_charts_summary_model

Create summary model from existing charts model

SYNTAX

status **gui_create_charts_summary_model**
-model *model_id*
[-max_rows *int*]
[-random]
[-sorted]
[-sort_column *int*]
[-sort_order *string*]
[-paged]
[-page_size *int*]
[-page_number *int*]

Data Types

model_id int

ARGUMENTS

-model *model_id*

Model to create summary model for.

-max_rows *int*

Maximum number of rows to extract from model.

-random

Select random rows.

-sorted

Use sorted rows.

-sort_column *int*

Specify sort column.

-sort_order *string*

Specify sort order.

-paged

Specify rows are paged.

-page_size ***int***

Specify page size (default 100).

-page_number ***int***

Specify page number.

DESCRIPTION

Create summary model from existing model.

EXAMPLES

The following example creates a summary of an existing model using random rows.

shell> gui_create_charts_summary_model -model $model -random

SEE ALSO

gui_create_charts_model(2)

gui_create_charts_text_annotation

Create text annotation in view or plot

SYNTAX

text_id **gui_create_charts_text_annotation**
{ -view *view_name* | -plot *plot_name* }
[-id *string*]
[-tip *string*]
{ -position *position* | -rectangle *bounding_box* }
-text *string*
[-properties *name_value_list*]

Data Types

view_name string
plot_name string
position string
bounding_box string
name_value_list string
text_id annotation_id

ARGUMENTS

-view *view_name*

Charts View to add annotation to.

-plot *plot_name*

Plot to add annotation to.

-id *string*

Optional identifier string for annotation.

-tip *string*

Optional tip string for annotation.

-position *position*

The position of the text.

-rectangle *bounding_box*

The bounding box of the text.

-text ***string***

The text to display.

-properties ***name_value_list***

Specifies a list of name value pairs for the properties to set on the created annotation.

RETURNS

text_id

Returns id of created text.

DESCRIPTION

Add a text annotation to plot or view.

The background, border and font of the text can be customized.

EXAMPLES

The following example creates a text annotation on a plot at (0 0) with the text "Test text".

```
shell> gui_create_charts_text_annotation -plot $plot -position {0 0} -text "Test text"
```

SEE ALSO

gui_create_charts_plot(2)
gui_create_charts_arrow_annotation(2)
gui_create_charts_ellipse_annotation(2)
gui_create_charts_point_annotation(2)
gui_create_charts_polygon_annotation(2)
gui_create_charts_polyline_annotation(2)
gui_create_charts_rectangle_annotation(2)
gui_remove_charts_annotation(2)
gui_set_charts_property(2)
gui_get_charts_property(2)

gui_create_clock_graph

Creates a clock graph.

SYNTAX

gui_create_clock_graph
-clct objects
[-levelized]
[-latency]

ARGUMENTS

-clct

Collection of objects to be used to generate a clock graph.

-levelized

Generates a levelized clock graph.

-latency

Generates a latency clock graph.

DESCRIPTION

The command generates a levelized or latency clock graph using a selected set of objects.

EXAMPLES

The following example generates a levelized clock graph using a collection of clock paths.

prompt> **gui_create_clock_graph -clct $clock_paths -levelized**

SEE ALSO

change_selection(2)

gui_create_clock_histogram

Create a histogram view for given clocks and corner.

SYNTAX

status **gui_create_clock_histogram**
-clocks *clocks*
-corner *corner*
-type *type*

Data Types

clock collection
corner collection
type string

ARGUMENTS

-clocks *clocks*

Specifies the clocks on which the latency histograms are calculated. This is a required option.

-corner *corner*

Specifies the corner from which the latency histogram is traced. This is a required option.

-type *type*

Specifies the latency histogram type. The option value could be one of "launch" or "capture". This is a required option.

DESCRIPTION

Create a new view to show the latency histogram of specified clocks at specified corner.

EXAMPLES

The following example create the launch latency histogram of clock CLK3 with corner cr1.

```
prompt> gui_create_clock_histogram \
  -clocks [get_clocks CLK3] -corner cr1 \
  -type launch
```

SEE ALSO

gui_create_menu

Create a menu item for the toplevel menubar or a context menu item. A menu hierarchy, based on the menu name, may be created as necessary to host the menu item.

SYNTAX

string **gui_create_menu** -menu *HierMenuName*
<-tcl_cmd *TclCmd*|-separator|-heading headingText>
[-enable_cmd *EnableCmd*]
[-get_state_cmd *GetStateCmd*]
[-icon *IconFilePath*]
[-hot_key *HotKey*]
[-tooltip *ToolTip*]
[-help_string *HelpStr*]
[-anchor_item *AnchorItem*]
[-anchor_offset *AnchorOffset*]
[-position *MenuPos*]
[-window_type *WindowTypeName*]
[-root *ContextMenuRoot*]
[-reference *RefMenuItem*]

HierMenuName *String*
TclCmd *String*
EnableCmd *String*
GetStateCmd *String*
IconFilePath *String*
HotKey *String*
ToolTip *String*
HelpStr *String*
AnchorItem *String*
AnchorOffset *Integer*
MenuPos *Integer*
WindowTypeName *String*
ContextMenuRoot *String*
RefMenuItem *String*

ARGUMENTS

-menu *HierMenuName*

HierMenuName specifies the hierarchical menu name of the menu item to create. If a level of the hierarchy does not exist in the menu structure it will be created automatically by this command. The "name" of a menu is a string which specifies the hierarchy of menu labels connected with '->'. For example, File->Quit specifies the Quit menu item under the File menu. These names are allowed to also specify the mnemonic for the menu text which is specified with an embedded '&', but they are not required to do

so.

-tcl_cmd *TclCmd*

TclCmd specifies the tcl command to execute when the menu item is selected. This option is mutually exclusive with the -separator and -heading options.

-separator

This option creates a menu item separator. It is mutually exclusive with the -tcl_cmd and -heading options.

-heading *headingText*

This option creates a text heading in the menu with the specified **headingText**. It is visually distinct from actual menu items, and it is not selectable like an actual menu item is. This is used to provide general grouping information without requiring an additional level of sub menu. This option is mutually exclusive with the -separator and -tcl_cmd options.

-enable_cmd *EnableCmd*

EnableCmd specifies the tcl command to evaluate whether the menu item should be enabled or disabled. The tcl command should return "true" or "1" to enable, and return "false" or "0" to disable. If the enable_cmd is not specified then the menu item will always be enabled.

-get_state_cmd *GetStateCmd*

GetStateCmd specifies the tcl command that will be executed to determine whether the menu item should be in an on (checked or pushed-in) or off (unchecked or normal) state. Most menu items don't have persistent state, and for those the get_state_cmd should not be specified. This tcl command returns "true" or "1" for the "on" state, and returns "false" or "0" for the "off" state.

-icon *conFilePath*

IconFilePath specifies the absolute pathname of the icon file. The file should be in *.xpm* format. Typical application-supplied icons are 16x16 pixels in size and use a transparent background (color 'None' in xpm format).

-hot_key *HotKey*

HotKey specifies the hotkey (accelerator) for the menu item.

-tooltip *ToolTip*

ToolTip specifies the tooltip for the menu item.

-help_string *HelpStr*

HelpStr specifies the help string for the menu item. The help string is displayed in the status bar.

-anchor_item *AnchorItem*

AnchorItem specifies the existing menu item that will be used as the anchor position for this new menu item. This option is used together with the -anchor_offset option to determine the location of the new menu item. The -position option cannot be used along with the -anchor_item and -anchor_offset options.

-anchor_offset *AnchorOffset*

AnchorOffset specifies the offset from the anchor item position that will be used to determine the location of the new menu item in the menu hierarchy. The offset must be a positive or negative integer, and not zero. If positive, the new menu item will be placed below the anchor item by the specified offset. If negative, it will be placed above the anchor item position by the absolute value of the specified offset. The -position option cannot be used along with the -anchor_item and -anchor_offset options.

-position *MenuPos*

MenuPos specifies the absolute menu position for the menu item within the parent popup menu. The -position option cannot be

used with the -anchor_item and -anchor_offset options.

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

-root *ContextMenuRoot*

ContextMenuRoot specifies the context menu root to which the menu item being defined will be added. A context menu root groups menu items that share the same context menu root. The context menu root is used for context menu. Context menu, is usually brought up by right clicking on a window and is used to provide context-sensitive features. (Note the difference between a context menu root and the menu root use by a toplevel window's menu bar .)

-reference *RefMenuItem*

RefMenuItem specifies the menu item under the toplevel window's menu bar, which will be used to define the behavior of the menu item being defined. Often used with the -root option to help associate a context menu item with an already defined menu entry under the menu bar. Used with the -window_type option to completely specify the reference menu item under the menu bar. If the -window_type option is not specified, the default toplevel window type as specified with the tcl variable "::gui_default_window_type" will be used.

DESCRIPTION

This command creates a menu item for the menu bar of a top level window. A menu hierarchy, based on the menu name, may be created as necessary to host the menu item. The menu hierarchy separator is denoted by "->".

These settings are stored either in a user-specific setup file or one that is shared across the installation. The gui initialization sequence will load builtin system configuration, and then apply the installation specific customizations, and finally the user's customizations.

The installation specific setup file is specified by the variable gui_custom_setup_file and will have a default empty value. This can be overridden in the .synopsys_dc.setup file if it is desired. If the file specified by this variable exists then it will be loaded after the gui is initialized.

The user setup file is ~/.synopsys_icc_gui.tcl for IC Compiler.

EXAMPLES

The following simple example adds a new toplevel popup menu *&Favorite* in the menubar and add a submenu named *Preferences* then add an item named *My &Item* to that submenu. This menu item will be enabled if *enable_my_item* is set to 1 and disabled otherwise. A hotkey and tooltip are also provided.

```
set enable_my_item 1
proc enable_my_item {} {
   if { $::enable_my_item == 1 } {
      return 1
   }
   return 0
}
gui_create_menu -menu "&Favorite->Preferences->My &Item" \
```

```
    -tcl_cmd "echo {executing My Item}" \
    -enable_cmd "enable_my_item" \
    -hot_key "Ctrl+N" \
    -tooltip "short tooltip description for My Item" \
    -help_string "long status bar description for My Item" \
    -icon_file "/syn/sparc/my_item.xpm"
```

The following IC Compiler example illustrates creating a new menu item in the context menu of the layout window which will refer to the Zoom Fit Selection menu item from the Layout window main menu bar. This assumes that the tcl variable "gui_default_window_type" is set to the ICC window type "LayoutWindow" and thus it is not necessary to specify the -window_type option with the value of "LayoutWindow".

```
gui_create_menu -menu "Zoom Fit Selection" \ -root layout_view_menu_root \ -reference "View->Zoom->Zoom Fit Selection"
```

The following example create a new Tcl-based menu entry in the context menu which invokes a procedure # called do_my_thing.

```
gui_create_menu -menu "Do My Thing" \ -root layout_view_menu_root \ -tcl "do_my_thing"
```

The following IC Compiler example illustrates adding a context menu item to the path schematic context menu which refers to a menu item in the Main window. This example assumes that there is a context menu root called "path_schematic_contextmenu_root", a hierarchical menu item in the toplevel menu bar of the window type "MainWindow" with the hierarchical menu name "Schematic->Delete Selected".

```
gui_create_menu -menu "Remove Selected" \ -root path_schematic_contextmenu_root \ -reference "Schematic->Delete Selected" -window_type MainWindow
```

SEE ALSO

gui_set_hotkey(2)
gui_report_hotkeys(2)
gui_delete_menu(2)
gui_create_toolbar(2)
gui_delete_toolbar(2)
gui_create_toolbar_item(2)
gui_delete_toolbar_item(2)
gui_get_menu_roots(2)
gui_default_window_type(3)
gui_custom_setup_files(3)

gui_create_message_waiver

Creates an EMS message waiver to waive matching messages in Message Browser.

SYNTAX

status **gui_create_message_waiver**
-name *waiver_name*
[-rule *EMS_check_rule_name*]
[-domain *EMS_domain_name*]
[-comment *comment_string*]
-filter *filter_list*

Data Types

waiver_name string
EMS_check_rule_name string
EMS_domain_name string
comment_string string
filter_list list

ARGUMENTS

-name *waiver_name*

Specifies the name of the waiver to create.

-rule *EMS_check_rule_name*

Specifies the name of EMS check rule whose error or warning messages will be matched against the filter. If not specified, the filter is applied to messages generated by all rules of a given domain, or on all messages of currently opened database, if no domain is specified. The default is empty.

-domain *EMS_domain_name*

Specifies the name of EMS domain whose error/warning messages will be matched against the filter. If not specified, the filter is applied to all messages of currently opened database. The default is empty.

-comment *comment_string*

Specifies the comments or notes to the waiver.

-filter *filter_list*

Specifies list of criteria for the filter. The criteria should contain column name, a match value, and match operation in list format {COULMN_NAME MATCH_OP MATCH_VALUE}. The column name should be exactly the same as visible param title defined in the EMS database for related checks, or the column name in message list of Message Browser. Each of matched values will be

converted to predefined EMS type for comparison, such as string, int, float, and so on. The possible matching operations are '==', '=~'(match), '<', '<=', '>', '>=' applied to suitable types. A filter can contain multiple criteria for different column fields. A messages is waived only when all criteria within this option are matched. This is a required option.

DESCRIPTION

The **gui_create_message_waiver** command creates a waiver with the filter list. The waiver is used to waive the matching EMS message from the Message Browser. The tool tests the messages against the filter strings defined in the waiver. When a string matches the criteria, the message status is set to 'waived'. If 'hide waived' preference is enabled, the waived messages are hidden from the message browser.

EXAMPLES

The following command creates waiver which will filter message generated by rule 'CTS_QOR_002' based on the filter list:

```
prompt> gui_create_message_waiver -name waiver_2 -rule CTS_QOR-002 \
  -filter { { Num <= 20 } { { Net Name } == gclk } { Capacitance == 0.45 } }
```

SEE ALSO

gui_remove_message_waivers(2)
gui_create_message_waiver(2)

gui_create_pref_category

Create a preference category.

SYNTAX

string **gui_create_pref_category** -category *Category*

Category *String*

ARGUMENTS

-category *Category*

Category specifies the category to create.

DESCRIPTION

This command creates a preference service category with the specified name, which can be used to store and categorize key-value pairs that are created with the **gui_create_pref_key** command. If successful, the command returns the name of the category that is created.

EXAMPLES

The following example create a user-defined category *some_category* for storing preferences and then create a boolean key *some_key* under that category with the initial value of false.

```
gui_create_pref_category -category some_category
gui_create_pref_key -category my_category -key some_key -value_type bool -value false
```

SEE ALSO

gui_create_pref_key(2)
gui_set_pref_value(2)
gui_get_pref_value(2)

gui_get_pref_value_type(2)
gui_remove_pref_key(2)
gui_get_pref_categories(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_create_pref_key

Create a preference key.

SYNTAX

string **gui_create_pref_key** -key *Key*
-value_type *ValueType*
-value *Value*
[-category *Category*]
[-keep_value_if_exist]
[-read_only]
[-description *Description*]
[-min *Minimum_Value*]
[-max *Maximum_Value*]
[-legal_value_list *Value_List*]
[-string_to_int_map *Value_Map*]
[-save_on_exit *Boolean_Value*]

Key *String*
Category *String*
ValueType *one of [bool|integer|double|color|string|rect|point|size]*
Value *Depends on the value type*
Description *String*
Minimum_value *Minimum integer or double value*
Maximum_value *Maximum integer or double value*
Value_List *String List*
Value_Map *A list of string pair lists, each mapping a string value to an integer value.*
Boolean_Value *true | false*

ARGUMENTS

-key *Key*

Key specifies the name of the key to create.

-value_type *ValueType*

ValueType specifies the data type of the value of the key. It must be one of [bool | integer | double | color | string].

-value *Value*

Value specifies the value of the key. The value should be set appropriately in accordance with its value type. The **bool** type accepts [true, false, 0, 1, on, off]. The **color** type accepts a named color, eg. "red" or a string specifying the color in this format

"#RRGGBB", for example, "#0000FF".

-category *Category*

Category specifies the category that the key belongs to. If not specified, a default category will be assigned.

-keep_value_if_exist

This Boolean option specifies to keep current value if the key already exists. The default is that the key will be assigned the value given in the **-value** option.

-read_only

If given, this flag specifies that the preference key value is read-only and its value cannot be set after the key has been created.

-description *Description*

If given, the given description string with the meaning of the key is assigned to the preference key.

-min *Minimum_Value*

If given, specifies the minimum value for an integer or double type preference key.

-max *Maximum_Value*

If given, specifies the maximum value for an integer or double type preference key.

-legal_value_list *Value_List*

If given, this option specifies discrete valid values for an iteger, double or string type preference key. Input the argument using Tcl list syntax: {{prefVal} ...}

-string_to_int_map *Value_Map*

If given, provides a map from a string key value to an integer value for preferences of integer type. Input the argument as a Tcl list of Tcl list where the internal list is a pair of the string value and the integer value: {{stringval intval} ...}

-save_on_exit *Boolean_Value*

If given, specifies whether the preference key value is saved to the user preference file upon existing the application. Acceptable argument is either **true** or **false**.

DESCRIPTION

This command creates a preference key with the specified key name. This key will have the specified value type and value, and it will be created under the specified category. If the category is not specified, the key will belong to the default category. Returns the value of the preference key.

EXAMPLES

The following example create a user-defined category *some_category* for storing preferences and then create an integer key *some_key* under that category with the initial value of 200.

```
gui_create_pref_category -category some_category
```

```
gui_create_pref_key -category my_category -key some_key -value_type integer -value 200
```

SEE ALSO

gui_create_pref_category(2)
gui_set_pref_value(2)
gui_get_pref_value(2)
gui_get_pref_value_type(2)
gui_remove_pref_key(2)
gui_get_pref_categories(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_create_schematic

Creates a schematic view.

SYNTAX

string **gui_create_schematic**
[-clct *objects*]
[-size *{w h}*]
[-be_specific *bool*]

Data Types

objects collection
{w h} string

ARGUMENTS

-clct *objects*

Generates a schematic for the specified object collection. The current selection is used if not specified.

-size *{w h}*

Generates a schematic for the specified width (w) and height (h). Application default view size is used if not specified.

-be_specific *bool*

Controls the behavior of the schematic generation as to whether additional objects beyond those specifed using -clct. If not specifed, or specifed as false, additional objects will be included in the new schematic to provide connectivity and context. If specified as true, only the objects specified will be included in the new schematic.

DESCRIPTION

The command generates a new schematic view from the current selection or an object collection if specified.

EXAMPLES

The following example generates a schematic view of the hierarchical cell named RAM_1.

```
prompt> gui_create_schematic -clct [find cell RAM_1]
```

SEE ALSO

change_selection(2)

gui_create_task

Create a task.

SYNTAX

gui_create_task
-name *TaskName*
-item_root *ItemRootName*
[-help_string] *HelpString*
[-menu_string] *MenuString*
[-icon] *IconFileName*
[-default]

TaskName *String*
ItemRootName *String*
HelpString *String*
MenuString *String*
IconFileName *String*

ARGUMENTS

-name *TaskName*

TaskName specifies the name of the new task. Task names must be unqiue.

-item_root *ItemRootName*

ItemRootName specifies the name of the task root that will be associated with the task. Item root names are unique. If the same item root is associated with multiple tasks, then the tasks share the same item root.

-help_string *HelpString*

HelpString specifies the string that will displayed as help to describe the task.

-menu_string *MenuString*

MenuString specifies the string that will be shown in a task selection menu for the task. If omitted, the task name is used.

-icon *IconFileName*

IconFileName specifies the name of the file with an icon definition for the icon to be displayed in the GUI for the task.

-default

This option specifies that this command will be the default task set as the current task. If another task is later created with this option, the the later setting overrides any previous settings.

DESCRIPTION

This command creates a task. A task defines the user task context for the whole application. Tasks may be used to configure menus and also to present a hierarchical list of task items to the user to navigate in the Task Assistant. The hierarhical list of task items is defined with the gui_create_task_item command.

EXAMPLES

The following simple example creates a new task, then defines a Tk widget task page, and an HTML task page, then creates task items referencing the task pages.

```
gui_create_task -name MyAppTask -item_root MyAppTaskRoot

gui_create_tk_task_page -name InputFormPage -create_command create_widget_proc
gui_create_html_task_page -name AppDocument -path ./MyAppDocument.html

gui_create_task_item -name "Application->Input Form" -task MyAppTask -page InputFormPage
gui_create_task_item -name "Application->Documentation" -task MyAppTask -page AppDocument
```

SEE ALSO

gui_create_task_item(2)
gui_create_tk_task_page(2)
gui_create_html_task_page(2)

gui_create_task_item

Create an item in the task tree to access in the Task Assistant.

SYNTAX

gui_create_task_item
-name *ItemName*
<-task *TaskName* | -item_root *ItemRootName*>
-page *PageName*
[-search_terms *TermsList*]

ItemName *String*
TaskName *String*
ItemRootName *String*
PageName *String*
TermsList *List*

ARGUMENTS

-name *ItemName*

ItemName specifies the hierarchical name given to the task item being created. This is also the text displayed to the user in the Task Assistant.

-task *TaskName*

TaskName specifies the task in which this task item is created. If a task name is used to create the task item, the item is inserted into the task item root associated with the task in the create_task command. If other tasks use the same item root, they will also see the new task item created with this command. This option is mutually exclusive with the -item_root option. One must be specified but not both.

-item_root *ItemRootName*

ItemRootName specifies the task item root in which this task item is created. Item root names are associated with task names with the gui_create_task command and allows multiple tasks to share a single task item root. This option is mutually exclusive with the -item_root option. One must be specified but not both.

-page *PageName*

PageName specifies the page content factory associated with this task item. When the user selects the task item, the page specified by this option will be constructed and shown in the Task Assistant.

-search_terms *TermsList*

TermsList specifies additional terms that can be matched by the command search feature to find this task item. For example, if the task page allows the user to specify values for input parameters to a command, you may want to associate the name of that

command as additional search term for the task item.

DESCRIPTION

This command creates a task item in the given task. Task items are defined in a hierarchy with hierarchy levels delimited by the separator string "->". The hierarchy of task items are shown in the task item navigation tree in the Task Assistant.

When the user selects a leaf item in the task navigation tree, the Task Assistant will show the task page associated with the item by this command. Currently, the pages can either be constructed from an HTML document or from a Tk widget factory.

EXAMPLES

The following simple example adds a new task, then defines a Tk widget task page, and an HTML task page, then creates task items referencing the task pages.

```
gui_create_task -name MyAppTask -item_root MyAppTaskRoot

gui_create_tk_task_page -name InputFormPage -create_command create_widget_proc
gui_create_html_task_page -name AppDocument -path ./MyAppDocument.html

gui_create_task_item -name "Application->Input Form" -task MyAppTask -page InputFormPage
gui_create_task_item -name "Application->Documentation" -task MyAppTask -page AppDocument
```

SEE ALSO

gui_create_task(2)
gui_create_tk_task_page(2)
gui_create_html_task_page(2)

gui_create_task_page

Creates a task page with Command Form, HTML content or Tk Command Form.

SYNTAX

gui_create_task_page
-name taskPageName
-format pageType
-source contentSource

Data Types

taskPageName string

pageType string

contentSource string

ARGUMENTS

-name *taskPageName*

taskPageName specifies the name of a new task page.

-format *pageType*

Define this new page as a specified Type Page. **pageType** can be command, html, tk or tab. Which specifies the Command Form Page, HTML Page, Tk Command Form Page or Tabs Page.

-source *contentSource*

Define the content to be shown in this page. If this page is a Command Form Page or a Tk Command Form Page, **contentSource** is the command name. If it is a HTML Page, **contentSource** specifies the full path to the file with HTML source for the task page. If it is a Tabs Page, **contentSource** specifies a list of existing page names or page&title pairs.

DESCRIPTION

The **gui_create_task_page** command creates several kinds of task pages which can be shown in the task assistant. The name option value is the unique name of the page. The format option defines which kind of page to be created and the source option defines what content to be shown in the task page.

EXAMPLES

The following simple example creates three new task pages, then show them in the task assistant.

```
prompt> gui_create_task_page -name page1 -format command -source gui_report_map
prompt> gui_create_task_page -name page2 -format html -source ./MyAppDocument.html
prompt> proc create_widget_proc {widgetName parentName} {
        p#populate $wigetName with Tk
       p}
prompt> gui_create_task_page -name page3 -format tk -source create_widget_proc
prompt> gui_create_task_page -name page4 -format tab -source {page1 page2 {page3 title} }
prompt> gui_create_task -task "Testing Task" -item_root test_task_item_root
prompt> gui_create_task_item -task "Testing Task" -name "Testing Page 1" -page page1
prompt> gui_create_task_item -task "Testing Task" -name "Testing Page 2" -page page2
prompt> gui_create_task_item -task "Testing Task" -name "Testing Page 3" -page page3
prompt> gui_create_task_item -task "Testing Task" -name "Testing Page 4" -page page4
prompt> gui_show_task -task "Testing Task:Testing Page"
```

SEE ALSO

gui_create_task(2)
gui_create_task_item(2)

gui_create_tk_palette_type

Create a type of Tk palette.

SYNTAX

status **gui_create_tk_palette_type**
-type *string*
-title *string*
[-icon *string*]
-window_types *string_list*
[-dock_edge *dock_edge*]
-create_command *string*

Data Types

dock_edge string

ARGUMENTS

-type *string*

Unique palette type name for the palette.

The type name must be non-empty, must start with a letter or underscore, and must only contain letter, number or underscore characters.

-title *string*

Title string for the palette.

-icon *string*

Icon to display for the palette.

-window_types *string_list*

List of window types which the palette is to be added to.

-dock_edge *dock_edge*

The default edge to add the palette

left - dock on left edge
right - dock on right edge
top - dock on top edge
bottom - dock on bottom edge

-create_command *string*

The option specifies the tcl command to be run when the palette is created so that the palette's tk widget can be populated with tk child widgets and any other initialization can be performed.

The tcl procedure takes two arguments 'windowName' and 'parentName' where 'windowName' is the tk palette widget to populate and 'parentName' is the name of the parent window.

DESCRIPTION

Creates a user defined tk palette when a window of any of the specified types is created.

EXAMPLES

The following example creates a palette of type 'mypalette', with a title of 'My Palette' which calls the 'build_my_palette' tcl procedure to populate the palette with tk widgets whenever a new window of type 'LayoutWindow' is created.

The 'build_my_palette' creates a text entry filed in a frame for the palette.

```
shell> proc build_my_palette {windowName parentName} {
  set top_frame [frame $windowName.frame]
  pack $top_frame -fill both -expand 1
  set entry [entry $top_frame.entry -textvariable $::var]
  pack $entry
}
shell> gui_create_tk_palette_type -type mypalette -title "My Palette" \
 -create_command build_my_palette -window_types "LayoutWindow" -dock_edge right
```

SEE ALSO

gui_execute_events(2)

gui_create_toolbar

Create a toolbar with the specified name.

SYNTAX

string **gui_create_toolbar** -name *ToolBarName*
[-title *Title*]
[-dock_side *DockSide*]
[-hidden]
[-window_type *WindowTypeName*]

ToolBarName *String*
Title *String*
DockSide *String*
WindowTypeName *String*

ARGUMENTS

-name *ToolBarName*

ToolBarName is the toolbar identifier for the new toolbar.

-title *Title*

Title specifies the title or caption of the toolbar.

-dock_side *DockSide*

DockSide specifies the side of the window the tool docks to by default. The legal values are top, bottom, left, right.

-hidden

-hidden specifies the toolbar should be hidden by default.

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

DESCRIPTION

This command creates a toolbar with the given name and title. Returns the name of the toolbar created.

EXAMPLES

Create a toolbar called mytoolbar and add a button to it that invokes the File->Quit menu item.

```
gui_create_toolbar -name mytoolbar
gui_create_toolbar_item -name mytoolbar -menu "File->Quit"
```

SEE ALSO

gui_set_hotkey(2)
gui_report_hotkeys(2)
gui_create_menu(2)
gui_delete_menu(2)
gui_delete_toolbar(2)
gui_create_toolbar_item(2)
gui_delete_toolbar_item(2)
gui_get_toolbar_names(2)
gui_default_window_type(3)
gui_custom_setup_files(3)

gui_create_toolbar_item

Create a button in the specified toolbar.

SYNTAX

string **gui_create_toolbar_item** -toolbar *ToolBarName*
<-menu *MenuName*|-separator *SeparatorName*>
[-window_type *WindowTypeName*]

ToolBarName *String*
MenuName *String*
SeparatorName *String*
WindowTypeName *String*

ARGUMENTS

-toolbar *ToolBarName*

ToolBarName specifies the toolbar that the new toolbar item will be added to.

-menu *MenuName*

MenuName specifies the name of the menu item to be invoked from this toolbar button. The "name" of a menu is a string whcih specifies the hierarchy of menu labels connected with '->'. For example, File->Quit specifies the Quit menu item under the File menu. These names are allowed to also specify the mnemonic for the menu text which is specified with an embedded '&', but they are not required to do so. This option is mutually exclusive with the -separator option.

-separator *SeparatorName*

SeparatorName specifies the unique separator name within the toolbar. This option is mutually exclusive with the -menu option.

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

DESCRIPTION

This command adds a toolbar button to the specified toolbar using information described in the -menu option. The -menu option specifies the hierarchical menu name for some menu item. Thus, the toolbar icon becomes another interface for a menu item. Returns the item name if successful.

EXAMPLES

Create a toolbar called mytoolbar and add a button to it that invokes the File->Quit menu item.

```
gui_create_toolbar -name mytoolbar
gui_create_toolbar_item -name mytoolbar -menu "File->Quit"
```

SEE ALSO

gui_set_hotkey(2)
gui_report_hotkeys(2)
gui_create_menu(2)
gui_delete_menu(2)
gui_create_toolbar(2)
gui_delete_toolbar(2)
gui_delete_toolbar_item(2)
gui_get_toolbar_names(2)
gui_default_window_type(3)
gui_custom_setup_files(3)

gui_create_utable

Create a new UserTable.

SYNTAX

string **gui_create_utable**
-name *Table_Name*
-parent *Table_Name*
-columns *Column_List*
-timing_path *Tim_Clct*
[-meta_obj *MetaObj_Name*]

Table_Name String
Column_List List of Strings
Tim_Clct Timing Path Collection
MetaObj_Name String

ARGUMENTS

-name *Table_Name*

The name of the user table to create.

-parent *Table_Name*

The name of the parent table that is related to this user table.

-columns *Column_List*

A TCL list of column names for the table with optional data type postfix information.

-timing_path *Tim_Clct*

Create and fill a Timing Point Table with the given Timing Path Collection. This is exclusive with the -columns option.

-meta_obj *MetaObj_Name*

Optional MetaData object name used to define table meta data. See the command gui_set_utable_meta for more information.

DESCRIPTION

This command creates a new user table with the given list of column names.
In addition, column names can have postfixes that describe the data type of that column.
Or one can use a MetaData object to define the column data types.

Important: This command returns the actual table name created which might
be different if the given table name had bad characters or already exists.
So, capture the return value and use it in future calls to this table.

Supported column name postfixes are:

:string
:int
:double
:cell
:net
:port
:pin
:point
:endpoint
:file

EXAMPLES

The following example creates a new UserTable with the given list of columns. The last
two columns also have some data type postfixes.

```
shell> set table [gui_create_utable -name my_table -columns {colA colB:int colC:double}]
```

SEE ALSO

gui_append_utable(2)
gui_change_selection_utable(2)
gui_close_utable(2)
gui_export_utable(2)
gui_get_utable(2)
gui_import_utable(2)
gui_set_utable_meta(2)
gui_open_utable(2)
gui_show_utable(2)
gui_write_utable(2)

gui_create_var

Create a typed gui variable.

SYNTAX

string **gui_create_var** -name *VarName*
-value *Value*
[-value_type *ValueType*]
[-keep_value_if_exist]
[-read_only]

VarName *String*
Value *Depends on the value type*
ValueType *one of [bool|integer|double|color|string|point|size|rect]*

ARGUMENTS

-name *VarName*

VarName specifies the name of the typed tcl variable to create.

-value_type *ValueType*

ValueType specifies the data type of the value of the variable. It must be one of [bool | integer | double | color | string]. If not specified, default to "string".

-value *Value*

Value specifies the value of the tcl variable. The value should be set appropriately in accordance with its value type. The **bool** type accepts [true, false, 0, 1, on, off]. The **color** type accepts a named color, eg. "red" or a string specifying the color in this format "#RRGGBB", for example, "#0000FF".

-keep_value_if_exist

If this option is specified, the variable will keep its current value if it already exists.

-read_only

If this option is specified, the variable will be read_only.

DESCRIPTION

This command creates a typed tcl variable with the specified name, value type and value. Returns the value of the variable created.

EXAMPLES

The following example creates a typed variable named *my_var* of type integer with value 200.

```
gui_create_var -name my_var -value_type integer -value 200
```

SEE ALSO

gui_set_var(2)
gui_get_var(2)
gui_remove_var(2)
gui_exist_var(2)

gui_create_vm

Creates a new visual mode container for visual mode buckets. The buckets contain coloring attributes for a collections of design objects.

SYNTAX

string **gui_create_vm**
-name *identifier*
[-update_cmd *update_command*]
[-title *label*]
[-help_topic *subject*]
[-infotip *infotip*]
[-discrete true | false]
[-icon *string*]
[-netfilter *string*]
[-float *bool*]
[-top_exaggeration *float*]
[-mid_exaggeration *float*]
[-bot_exaggeration *float*]
[-show_only_pins_of_nets *bool*]
[-enable_bucket_delete *bool*]

Data Types

identifier string
update_command string
label string
subject string
infotip string
net_filter string
icon_spec string

ARGUMENTS

-name *identifier*

Specifies the name that identifies the visual mode. Use this name when another visual mode command requires a visual mode name.

-update_cmd *update_command*

Specifies a Tcl command to execute when the visual mode is accessed. Use this option to update the state of the visual mode if its contents are likely to change under certain circumstances.

-title *label*

Specifies an optional title that is used to label the visual mode in the GUI. The *label* string can contain space characters. By default, the title is the empty string, and the **-name** option value is used to provide the label.

-help_topic *subject*

Specifies the help topic text string. This text is used as the subdirectory to find a help page related to the visual mode.

-infotip *infotip*

Specifies the infoTip text string.

-discrete true | false

Specifies an optional true or false string indicating whether the buckets of this visual mode are discrete or continuous. Only discrete buckets can be reordered. By default, visual modes are continuous.

-icon *string*

Specifies the icon file. The file contains an icon image for the GUI menus.

-netfilter *string*

Specifies the net connection filtering.

-float true | false

Specifies whether the bucket contents have a floating point range. By default, this option is false.

-top_exaggeration *float*

Specifies the exaggeration for the top bucket. The value should be greater than or equal to zero. By default, the value is 0.

-mid_exaggeration *float*

Specifies the exaggeration for the middle bucket. The value should be greater than or equal to -1. By default, the value is -1.

-bot_exaggeration *float*

Specifies the exaggeration for the bottom bucket. The value should be greater than or equal to zero. By default, the value is 0.

-show_only_pins_of_nets true | false

Specifies whether the layout shows pins of net objects in buckets. By default, the value is false.

-enable_bucket_delete true | false

Specifies whether the buckets can be deleted from context menu. Only discrete visual mode buckets can be deleted. By default, the value is false.

DESCRIPTION

The **gui_create_vm** command creates an instance of a visual mode. The visual mode provides a container for visual mode buckets, which associate coloring attributes with collections of design objects.

If the -update_cmd that you specify needs input from the user, then you can define the arguments to your procedure with define_proc_attributes, and then use the gui_show_command_form command for your procedure as the -update_cmd. This will show a form that will prompt for the arguments to your procedure and then it will call your proc to update the visual mode using those values. See the man pages for gui_show_command_form and define_proc_attributes for additional details.

EXAMPLES

The following example creates a new visual mode named mycoloring1 with the GUI label "Cells Colored By X":

```
prompt> gui_create_vm -name mycoloring1 -title {Cells Colored By X}
```

SEE ALSO

gui_create_vmbucket(2)
gui_get_vm(2)
gui_get_vmbucket(2)
gui_remove_vm(2)
gui_remove_vmbucket(2)
gui_set_vm(2)
gui_set_vmbucket(2)
gui_update_vm(2)
gui_show_command_form(2)
define_proc_attributes(2)

gui_create_vm_objects

Create objects to hold annotations in visual modes

SYNTAX

string **gui_create_vm_objects** *<clct>*

string *<clct>*

ARGUMENTS

<clct>

Objects to convert.

DESCRIPTION

Layout visual modes are used to color objects in the layout given certain criteria. Sometimes you want not only object coloring but some additional annotations displayed with the objects in a bucket.

In order to create this type of visual mode, you need special type of object that can hold annotations. Use this command to create these special objects.

Objects of this type support name and core_obj attribute. The core_obj attribute returns the object that was used to create this object.

EXAMPLES

shell> **set objs [gui_create_vm_objects $cells]**

SEE ALSO

gui_create_vm(2)
gui_create_vmbucket(2)
gui_update_vm_annotations(2)

gui_create_vmbucket

Create new Visual Mode Bucket

SYNTAX

string **gui_create_vmbucket** -vmname *mode_identifier*
-name *bucket_identifier*
[-infotip *infotip*]
[-netfilter *net_filter*]
[-color *color*]
[-pattern *pattern*]
[-exaggeration double]
[-number int]
[-maxval double]
[-minval double]
[-visible *visibility*]
[-title *label*]
[-collection *handle*]
[-above *bucket_identifier* | -below *bucket_identifier* -at *top|bottom*]

string *mode_identifier*
string *bucket_identifier*
string *infotip*
string *net_filter*
string *color*
string *pattern*
string *visibility*
string *label*
string *handle*
string *bucket_identifier*
string *bucket_identifier*
string *top|bottom*

ARGUMENTS

-vmname *mode_identifier*

Specifies the visual mode to which the bucket is a member. The visual mode must already exist. To see the current list of defined visual modes use the gui_list_vm command.

-name *bucket_identifier*

Specifies the name by which the visual mode bucket will be identified. It is used as the argument when associated visual mode commands require a visual mode bucket name to be specified.

-infotip *infotip*

String for infotip.

-netfilter *net_filter*

Specifies net connection filtering.

-color *color*

Specifies bucket rendering color. Supports color naming by RGB triplet (for example: "#FFFF00" for yellow) as well as standard names (for example: "yellow").

-pattern *pattern*

Specifies bucket rendering fill pattern. Supported patterns are: SolidPattern, HorPattern, VerPattern, CrossPattern, BDiagPattern, FDiagPattern, DiagCrossPattern, Dense1Pattern, Dense2Pattern, Dense3Pattern, Dense4Pattern, Dense5Pattern, Dense6Pattern, Dense7Pattern, NoBrush.

-exaggeration double

Specifies bucket min pixel exaggeration value >= 0.

-number int

Specifies bucket display number value >= 0 that is used to provide a display number for the visual mode bucket in the gui. If the -number option is not specified, it will default to the number of objects in the bucket.

-maxval double

Specifies bucket maximum value.

-minval double

Specifies bucket minimum value.

-visible *visibility*

Specifies initial visibility of bucket. Valid values are TRUE and FALSE.

-title *label*

Specifies an optional string (that can include embedded spaces) that is used to provide a label for the visual mode bucket in the gui. If the -title option is not specified, it will default to an empty string and the -name argument value value will be used to provide the labeling instead.

-collection *handle*

Tcl object collection to be colored by this visual mode bucket.

-above *bucket_identifier*

Specifies an optional visual mode bucket name above which the current bucket is to be rendered.

-below *bucket_identifier*

Specifies an optional visual mode bucket name below which the current bucket is to be rendered.

-at *top*|*bottom*

Specifies an optional string indicating whether the current visual mode bucket is to be rendered at the top or bottom of all other buckets. Valid values are top and bottom.

DESCRIPTION

The gui_create_vmbucket command creates an instance of a visual mode bucket. The visual mode bucket is a member of a specific visual mode. The visual mode bucket can be assigned coloring styles and a collection of design objects to which the coloring will be applied when they are rendered as a part of the visual mode.

EXAMPLES

Create a visual mode with two buckets, one whose objects will be colored red, and another whose objects will be colored blue:

shell> **gui_create_vm -name foo1 -title "My Visual Mode"**

shell> **gui_create_vmbucket -vmname foo1 -name red -title "Red" -color red**

shell> **gui_create_vmbucket -vmname foo1 -name blue -title "Blue" -color blue -below red**

SEE ALSO

gui_create_vm(2)
gui_get_vm(2)
gui_get_vmbucket(2)
gui_list_vm(2)
gui_remove_vm(2)
gui_remove_vmbucket(2)
gui_set_vm(2)
gui_set_vmbucket(2)
gui_update_vm(2)

gui_create_window

Creates a window of the specified window type.

SYNTAX

string **gui_create_window**
-type *window_type*
[-parent *parent_window_id*]
[-show_state *window_state*]
[-title *title*]
[-rect *rect* | -size {*width height*}]
[-icon *icon*]

Data Types

window_type string
parent_window_id string
window_state string
title string
rect rectangle
width integer
height integer
icon string

ARGUMENTS

-type *window_type*

Specifies the type of window you are creating. The tool creates an instance of this window type.

You can display a list of the supported window types by using the **gui_get_window_types** command.

-parent *parent_window_id*

Specifies the window ID of the parent top-level window for the view window you are creating.

If you do not specify a window ID and the window type is not a top-level window type, the tool uses the current top-level window. If there is no current top-level window, the tool creates a new top-level window automatically to host the window you are creating.

-show_state *create_window_state*

Specifies the window state in which to display the window you are creating. The result varies depending on whether the window is a top-level window or a view window.

For a top-level window, you can specify one of the following states:

maximized -- show maximized (fill whole screen)
minimized -- show iconified
normal -- show at preferred size

Note that the window manager might not fully honor the specified state. See your window manager documentation for more details.

For a view window, you can specify one of the following states:

maximized -- show maximized in view area and raise to top
minimized -- show iconified in view area
normal -- show at preferred size in view area and raise to top

If you display a view window in its maximized state, the tool also maximizes any existing view windows. If you display a view window in its minimized or normal state, the tool changes any maximized view windows to their normal states.

The default is -show_state normal.

-title *title*

Specifies the title for the top-level or view window you are creating.

Note that the window manager might not fully honor the specified top-level window title string. See your window manager documentation for more details.

-rect *rect*

Specifies the size and position of the window in the following format:

{{*x1 y1*} {*x2 y2*}}

The coordinates {*x1 y1*} specify the location of the top-left corner, and the coordinates {*x2 y2*} specify the location of the bottom-right corner. These coordinates are relative to the top-left corner of the screen, for a top-level window, or to the top-right corner of the view area in the parent top-level window, for a view window.

Note that the window manager might not fully honor the specified top-level window geometry. See your window manager documentation for more details.

The **-rect** option and the **-size** option are mutually exclusive.

-size {*width height*}

Specifies the width and height of the window.

Note that the window manager might not fully honor the specified top-level window geometry. See your window manager documentation for more details.

The **-size** option and the **-rect** option are mutually exclusive.

-icon *icon*

Specifies the image to be used for the top-level or view window icon.

For a view window, the icon appears in the top-left corner of the title bar when the window is in its normal or minimized state and at the left side of the menu bar when the window is maximized.

For a top-level window, the icon might not be displayed when the window is in its normal, minimized, or maximized state. See your window manager documentation for more details.

DESCRIPTION

This command creates a window of the specified type and returns the window ID. The window type that you specify controls whether the window is a top-level window or a view window.

EXAMPLES

The following example creates a top-level layout window and a minimized layout view window.

```
prompt> set top [gui_create_window -type LayoutWindow]
LayoutWindow.1
prompt> gui_create_window -type Layout -parent $top \
  -show_state minimized
layout.1
```

SEE ALSO

gui_close_window(2)
gui_exist_window(2)
gui_show_window(2)
gui_get_window_types(2)
gui_get_window_ids(2)
gui_set_active_window(2)
gui_get_current_window(2)

gui_define_charts_proc

Define tcl command to use in charts expression evaluation

SYNTAX

status **gui_define_charts_proc**
name
args
body

Data Types

ARGUMENTS

name

Name of the procedure.

The procedure name must be non-empty, start with an underscore or a letter and contain only underscore or letter characters.

args

The arguments of the procedure.

body

The body of the procedure.

DESCRIPTION

Define custom tcl command to use in charts expression evaluation.

The syntax and arguments are the same as the standard tcl **proc** command.

If the arguments and body strings are empty then any existing procedure of that name is deleted.

To list all the defined charts procedures you can use the command:

shell> gui_get_charts_data -name procs

To get the args and body for an existing procedure you can use the command:

shell> gui_get_charts_data -name proc_data -data <proc_name>

EXAMPLES

The following example changes a model to create a new column with the square root of values in the first column.

```
shell> gui_define_charts_proc test_proc { arg } { return [expr {$arg/2.0}] }
shell> gui_change_charts_model -model $model -add -expr "test_proc(column(0))" -type real
```

SEE ALSO

gui_change_charts_model(2)
gui_get_charts_data(2)

gui_delete_attrgroup

Removes a group of attributes or all attribute groups for an object type.

SYNTAX

status **gui_delete_attrgroup**
-class *design_object*
-name *name*
[-all]

Data Types

design_object string
name string

ARGUMENTS

-class *design_object*

Specifies the type of design object.

-name *name*

Specifies the name of the attribute group to be removed for the specified object type.

-all

Removes all attribute groups defined for the specified object type.

DESCRIPTION

The **gui_delete_attrgroup** command removes the specified attribute group or all attribute groups for the specified object type.

EXAMPLES

prompt> **gui_delete_attrgroup -class Cell -name "Hierarchy"**

prompt> **gui_delete_attrgroup -class Cell -all**

SEE ALSO

gui_create_attrgroup(2)
gui_list_attrgroups(2)
gui_update_attrgroup(2)

gui_delete_menu

Delete a menu item for the toplevel menubar or a context menu

SYNTAX

string **gui_delete_menu**
[-menu *HierMenuName* | -all]
[-window_type *WindowTypeName* | -root *ContextMenuRoot*]

HierMenuName *String*
WindowTypeName *String*
ContextMenuRoot *String*

ARGUMENTS

-menu *HierMenuName*

HierMenuName specifies the hierarchical menu name of the menu item to be deleted. The "name" of a menu is a string which specifies the hierarchy of menu labels connected with '->'. For example, File->Quit specifies the Quit menu item under the File menu. These names are allowed to also specify the mnemonic for the menu text which is specified with an embedded '&', but they are not required to do so. This option is mutually exclusive with the -all option.

-all

This option flag specifies all menu items in a menu root to be deleted. The option cannot be given with the -window_type option.

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

-root *ContextMenuRoot*

ContextMenuRoot specifies the context menu root to from which the menu item will be added. A context menu root name groups menu items to be included in a pop-up context menu. A context menu is usually brought up by right clicking on a window and is used to provide context-sensitive features. (Note the difference between a context menu root and the menu root use by a toplevel window's menu bar .)

DESCRIPTION

This command deletes a menu item for the menu bar of a top level window. The menu hierarchy separator is denoted by "->". If there

is a toolbar button or hotkeys defined for the menu being deleted they will also be deleted. If the path to the menu is a submenu that menu item and all its sub-items will be deleted.

The option -all may be used with a menu root name to remove all menu items from a menu. This is useful if a user wishes to customize or modify an entire context menu.

These settings are stored either in a user-specific setup file or one that is shared across the installation. The gui initialization sequence will load builtin system configuration, and then apply the installation specific customizations, and finally the user's customizations.

The installation specific setup file is specified by the variable gui_custom_setup_file and will have a default value of $::synopsys/admin/setup/.synopsys_galileo_gui.tcl. This can be overridden in the .synopsys_dc.setup file if it is desired. If the file specified by this variable exists then it will be loaded after the gui is initialized.

The user setup file is ~/.synopsys_galileo_gui.tcl.

EXAMPLES

The following example deletes the Quit item from the File menu.

```
gui_delete_menu -menu "File->Quit"
```

SEE ALSO

gui_set_hotkey(2)
gui_report_hotkeys(2)
gui_create_menu(2)
gui_create_toolbar(2)
gui_delete_toolbar(2)
gui_create_toolbar_item(2)
gui_delete_toolbar_item(2)
gui_default_window_type(3)
gui_custom_setup_files(3)

gui_delete_toolbar

Delete a toolbar with the specified name.

SYNTAX

string **gui_delete_toolbar** -name *ToolBarName*
[-window_type *WindowTypeName*]

ToolBarName *String*
WindowTypeName *String*

ARGUMENTS

-name *ToolBarName*

ToolBarName is the toolbar identifier for the toolbar to be deleted.

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

DESCRIPTION

This command deletes a toolbar and all of its toolbar items.

EXAMPLES

Delete a toolbar called mytoolbar.

```
gui_delete_toolbar -name mytoolbar
```

SEE ALSO

gui_set_hotkey(2)
gui_report_hotkeys(2)
gui_create_menu(2)
gui_delete_menu(2)
gui_create_toolbar(2)
gui_create_toolbar_item(2)
gui_delete_toolbar_item(2)
gui_get_toolbar_names(2)
gui_default_window_type(3)
gui_custom_setup_files(3)

gui_delete_toolbar_item

Remove a toolbar button specified by the menu name from the specified toolbar.

SYNTAX

string **gui_delete_toolbar_item** -toolbar *ToolBarName*
<-menu *MenuName*|-separator *SeparatorName*>
[-window_type *WindowTypeName*]

ToolBarName *String*
MenuName *String*
SeparatorName *String*
WindowTypeName *String*

ARGUMENTS

-toolbar *ToolBarName*

ToolBarName specifies the toolbar to be modified.

-menu *MenuName*

ItemName specifies the hierarchical menu name for some menu item, that this toolbar item is associated with. This option is mutually exclusive with the -separator option.

-separator *SeparatorName*

SeparatorName specifies the unique separator name within the toolbar. This option is mutually exclusive with the -item option.

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

DESCRIPTION

This command deletes a toolbar item specified by *MenuName* or *SeparatorName* from the specified toolbar. Returns the name of the deleted toolbar item if successful.

EXAMPLES

Delete the button bound to File->Quit from the toolbar mytoolbar.

```
gui_delete_toolbar_item -name mytoolbar -menu "File->Quit"
```

SEE ALSO

gui_set_hotkey(2)
gui_report_hotkeys(2)
gui_create_menu(2)
gui_delete_menu(2)
gui_create_toolbar(2)
gui_delete_toolbar(2)
gui_create_toolbar_item(2)
gui_get_toolbar_names(2)
gui_default_window_type(3)
gui_custom_setup_files(3)

gui_edit_vmbucket_contents

Edit the collection contents of a Visual Mode Bucket

SYNTAX

string **gui_edit_vmbucket_contents** -vmname *mode_identifier*
-name *bucket_identifier*
[-add | -remove | -replace]
[-collection *handle*]

string *mode_identifier*
string *bucket_identifier*
string *handle*

ARGUMENTS

-vmname *mode_identifier*

Specifies the visual mode to which the bucket is a member. The visual mode must already exist. To see the current list of defined visual modes use the gui_list_vm command.

-name *bucket_identifier*

Specifies the name of the visual mode bucket to be modified. To see the list of buckets associated with a specific visual mode use the gui_get_vm command with the -buckets option.

-add

Add the collection to the collection already in the bucket

-remove

Remove the collection from the collection already in the bucket

-replace

Replace the collection already in the bucket

-collection *handle*

Tcl object collection to be colored by this visual mode bucket.

DESCRIPTION

The gui_edit_vmbucket_contents command modifies the collection stored in a visual mode bucket. The collection can either be appended to, removed, or replaced. The flags to control this are mutually exclusive.

EXAMPLES

Add the current selection to the collection in the bucket 0 of the SNAPSHOT visual mode:

```
shell> gui_edit_vmbucket_contents -vmname SNAPSHOT -name 0 -add -collection [get_selection]
```

SEE ALSO

gui_create_vm(2)
gui_create_vmbucket(2)
gui_get_vm(2)
gui_get_vmbucket(2)
gui_list_vm(2)
gui_remove_vm(2)
gui_remove_vmbucket(2)
gui_set_vm(2)
gui_update_vm(2)

gui_error_browser

Show or hide the Error Browser Dialog

SYNTAX

string **gui_error_browser**
[-hide | -show | -toggle]

ARGUMENTS

-hide

Hides the Error Browser Dialog.

-show

Shows the Error Browser Dialog. If no option flag is provided, this is the default.

-toggle

Show the Error Browser Dialog if it is currently hidden, else hide it.

DESCRIPTION

The **gui_error_browser** command controls the visibility of the Error Browser Dialog. The Error Browser Dialog displays violations with physical shapes recorded by a checker, for example a DRC engine. This command is valid only when there is a design open. If the Error Browser Dialog is shown and it was not previously shown. If the Error Browser Dialog is shown and it was previously shown, it will show the error data last loaded.

EXAMPLES

The following example creates and shows the Error Browser Dialog:

```
gui_error_browser -show
```

SEE ALSO

gui_eval_cmd

Execute and (optionally) log a Tcl command.

SYNTAX

string **gui_eval_cmd** -cmd *command*
[-log | -no_log]
[-history | -no_history]
[-display| -no_display]
[-exec_callbacks | -no_exec_callbacks]

string *command*

ARGUMENTS

-cmd *command*

The command to execute and optionally log.

-log

Enables logging of the command. If neither -log nor -no_log is specified, -log is the default.

-no_log

Disables logging of the command.

-history

Enables putting the command into the command history. if neither -history nor -no_history is specified, the default is -history if logging is enabled, and -no_history if logging is disabled.

-no_history

Disables putting the command into the command history.

-display

Enables display of the command result. If neither -display nor -no_display is specified, -no_display is the default.

-no_display

Disables display of the command result.

-exec_callbacks

Enables running of shell execution callbacks for the command. If neither -exec_callbacks nor -no_exec_callbacks is specified, -no_exec_callbacks is the default.

-no_exec_callbacks

Disables running of shell execution callbacks for the command.

DESCRIPTION

Execute and (optionally) log a Tcl command.

The purpose of this command is to support logging of core commands run by Tcl code in qTcl dialogs. (It is just a wrapper around EkShellExecution::executeSync which exposes several of the executeSync boolean parameters.)

All options except -cmd are optional - the default behavior for the optional options is: -log -history -no_display -no_exec_callbacks.

For history, if neither -history nor -no_history are specified, the default is -history if logging is enabled, and -no_history if logging is disabled.

The command is hidden.

gui_eval_cmd will result in a "nested" command execution when run from the command line: the outermost command execution is the execution of gui_eval_cmd itself, and the nested command execution is for the -cmd command. A recursive call to EkShellExecution::executeSync will happen.

The result of the nested command is passed back as the result of gui_eval_cmd.

Some of the options, such as -exec_callbacks and -display, are included for completeness of the executeSync wrapper, and may not be very useful.

EXAMPLES

```
psyn_shell-t> gui_eval_cmd -cmd {echo abc}
abc
```

SEE ALSO

gui_log_cmd(2)

gui_eval_command

Executes and optionally logs a tool command language (Tcl) command.

SYNTAX

string **gui_eval_command**
-command *command*
[-echo]
[-history]
[-honor_preview | -preview]

Data Types

command string

ARGUMENTS

-command *command*

Specifies the Tcl command to execute and to record in the command replay log.

-echo

Echoes the command and displays the command result in the console log view. By default, the tool does not echo the command or display the result in the log view.

-history

Appends the command to the command history list in the console history view. By default, the tool does not append the command to the command history list.

-honor_preview

This option is mutually exclusive with the -preview option. Honors the Preview Command Only settting in the dialog. By default, the tool does not honor the Preview Command Only settting.

-preview

This option is mutually exclusive with the -honor_preview option. Runs the given command in preview mode, irregardless of the current preview mode set from a command dialog. When -preview is given, -echo is turned on since the previewed command cannot be seen in the xterm if -echo is turned off. As with all commands issued from command dialogs when in preview mode, if a script editor is present, then the previewed command is redirected to the scripte editor. Using this option does not set or clear the global preview mode in the application.

DESCRIPTION

This command executes a Tcl command and records it in the command replay log. Use the **-echo** option to echo the command and its result in the console log view Use the **-history** option to append the command to the command history list. Use the **-honor_preview** option to honor the Preview Command Only settting. Use the **-preview** option to preview the command only without executing it.

The result of a nested command is passed back as the result of **gui_eval_command**.

EXAMPLES

The following example evaluates the command "echo abc".

```
prompt> gui_eval_command -command {echo abc}
abc
```

The following example previews the command "echo abc".

```
prompt> gui_eval_command -command {echo abc} -preview
abc
```

gui_eval_task_command

Executes a command from a task assistant page

SYNTAX

string **gui_eval_task_command**
-command *command*
[-task taskName | -script]

Data Types

command string
command taskName

ARGUMENTS

-command *command*

Specifies the Tcl command to execute and to add to the favorites and most-recently-used (MRU) palettes. The executed command is logged in the command log.

-task *taskName*

If this option is present, then the given task name is used when the command is added to the favorites and MRU palettes. If not present, then the current task name is used.

-script

This option is mutually exclusive with the -task option. If this option is present, the command is not run but is appended in the script editor.

DESCRIPTION

This command executes the given Tcl command and records it in the command replay log. Use the **-script** option to append the command to the script editor only without executing it.

The result of a nested command is passed back as the result of **gui_eval_task_command**.

EXAMPLES

The following example evaluates the command "echo abc".

```
prompt> gui_eval_task_command -command {echo abc}
abc
```

The following example adds the command "echo abc" to the script editor.

```
prompt> gui_eval_task_command -command {echo abc} -script
abc
```

gui_execute_menu_item

Execute a menu item in the currently active window.

SYNTAX

status **gui_execute_menu_item**
{ -menu *menu_item_id* }

Data Types

menu_item_id string

ARGUMENTS

-menu *menu_item_id*

menu_item_id specifies the hierarchical name of the menu item to execute. The "name" of a menu is a string which specifies the hierarchy of menu labels connected with '->'. For example, File->Quit specifies the Quit menu item in the File menu. These names are allowed to also specify the keyboard accelerator for the menu text which is specified with an embedded '&', but they are not required to do so.

DESCRIPTION

This command checks for the existence and enabled state of the given menu item in the currently active window. If it exists and is enabled, the menu item is executed.

The command returns and result value string from the executed menu item, if any.

EXAMPLES

The following example activates the main window named Toplevel.2, then executes the **Hierarchy Browser** menu item in the **View** menu in Tolevel.2.

shell> gui_set_active_window -window Toplevel.2
shell> gui_execute_menu_item -menu "View->Hierarchy Browser"

SEE ALSO

gui_create_menu(2)
gui_set_active_window(2)

gui_exist_pref_category

Check the existence of a preference category.

SYNTAX

bool **gui_exist_pref_category** -category *Category*

Category *String*

ARGUMENTS

-category *Category*

Category specifies the category.

DESCRIPTION

This command checks the existence of a preference category. Return "1" if the specified category exists, otherwise return "0".

SEE ALSO

gui_create_pref_category(2)
gui_create_pref_key(2)
gui_set_pref_value(2)
gui_get_pref_value(2)
gui_get_pref_value_type(2)
gui_remove_pref_key(2)
gui_get_pref_categories(2)
gui_get_pref_keys(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_exist_pref_key

Check the existence of a preference key.

SYNTAX

bool **gui_exist_pref_key** -key *Key*
[-category *Category*]

Key *String*
Category *String*

ARGUMENTS

-key *Key*

Key specifies the preference key of interest.

-category *Category*

Category specifies the category. Default category will be used if missing.

DESCRIPTION

This command checks the existence of a preference key given the key name and the preference category. Return "1" if the specified key exists, otherwise return "0".

SEE ALSO

gui_create_pref_category(2)
gui_create_pref_key(2)
gui_set_pref_value(2)
gui_get_pref_value(2)
gui_get_pref_value_type(2)
gui_remove_pref_key(2)
gui_get_pref_categories(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)

gui_update_pref_file(2)

gui_exist_var

Check the existence of the typed tcl variable that is created with gui_create_var.

SYNTAX

bool **gui_exist_var** -name *VarName*

VarName *String*

ARGUMENTS

-name *VarName*

VarName specifies the variable name.

DESCRIPTION

This command checks the existence of the typed tcl variable with the specified name. Return "1" if the variable exists, otherwise return "0".

SEE ALSO

gui_create_var(2)
gui_set_var(2)
gui_get_var(2)
gui_remove_var(2)

gui_exist_window

Check for the existence of specified window or window type

SYNTAX

status **gui_exist_window**
{ -window *window_id* | -type *window_type* }
[-parent *window_id*]

Data Types

window_id string
window_type string

ARGUMENTS

-window *window_id*

Specifies the window id to test existence for.

This option is mutually exclusive with the *-type* option.

-type *window_type*

Specifies that the existence of a window of this window type is tested for.

If the *-parent* option is specified then the existence of a window of this window type is tested for in the specified toplevel window.

If the *-parent* option is **not** specified then the existence of a window of this window type is tested for in any toplevel window.

This option is mutually exclusive with the *-window* option.

-parent *window_id*

Specifies the toplevel window to check for existence in.

Note: This option is only relevant if the *-type* option is specified.

DESCRIPTION

This command checks the existence of the specified window, or window type across toplevel windows or within a specified toplevel window.

The command returns "1" or "0" for success or failure respectively.

EXAMPLES

The following example will create a toplevel window and a child layout view window, then check the existence of the command layout window.

```
shell> set top [gui_create_window -type TopLevel]
shell> set x [gui_create_window -type Layout -parent $top]
shell> set layout_exist [gui_exist_window $x]
shell> if { $layout_exist == 1} {echo "layout exist" }
```

SEE ALSO

gui_close_window(2)
gui_create_window(2)
gui_show_window(2)
gui_get_window_types(2)
gui_get_window_ids(2)
gui_set_active_window(2)
gui_get_current_window(2)

gui_explore_logic_hierarchy

Creates, expands, colors, and removes enclosing module boundaries for cells in the logical hierarchy. This command is typically called by the tool from the Color by Hierarchy section of the Hierarchy Exploration panel.

SYNTAX

module_boundaries **gui_explore_logic_hierarchy**
[-threshold *threshold_ratio*]
[-create | -expand | -collapse | -remove | -update | -query]
[-color *color_id*]
[-force]
[*cells*]

Data Types

module_boundaries string
threshold_ratio real
color_id integer
cells collection

ARGUMENTS

-threshold *threshold_ratio*

Specifies the ratio to use when determining if a child hierarchical cell should have an enclosing module boundary created for it. Module boundaries are created if the size of the boundary relative to the total size is greater than *threshold_ratio*. For example, if the *threshold_ratio* is 0.03, logical hierarchies are ignored if they have a size smaller than 3 percent of total size.

By default, the ratio is specified by the *plan.explore.default_threshold* application option.

-create

Creates enclosing module boundaries for the specified logical hierarchical cells. Module boundaries are removed for the child cells of the specified logical hierarchical cells.

If the **-color** option isn't specified, a color will be automatically assigned.

-expand

Creates enclosing module boundaries for the child cells of the logical hierarchical cells (limited by a size threshold). If the parent cell already has a module boundary then, the module boundary for the parent will be removed. If module boundaries already exist on the child modules of the cell, the boundaries are re-created or removed as needed. For example, boundaries could be removed of the threshold changed.

The tool automatically assigns a color to the cells of each created module boundary.

-collapse

Removes the enclosing module boundaries for the currently selected module boundary and any sibling module boundaries, then creates a parent module boundary.

-remove

Removes the selected enclosing module boundary.

-update

Updates the shape of the enclosing module boundaries for the currently selected shape based on the current placement of their child cells.

If no cells are specified, all enclosing module boundaries are updated.

The application keeps track of the state of the created enclosing module boundaries, so updating update on module boundaries with no cell location changes will not normally modify the module boundary. To force an update, use the **-force** option.

The color of the cells within the module boundary will not be changed.

-query

Returns the instances for the existing enclosing module boundaries.

-color *color_id*

Specifies a color ID for the module boundary and its descendant. Valid *color_id* values are integer values from 0 to 63. If this option isn't specified, module boundaries and its descendant will be colored automatically. When creating the boundary, if *color_id* is 0, do not color the cells; Otherwise, color the objects with *color_id*.

-force

Forces update of enclosing module boundaries, even if the tool considers them up-to-date.

This option is not typically used, but can be added if you determine that an update is needed.

This option can only be used with the **-update** option.

DESCRIPTION

This command supports hierarchy exploration by creating, expanding, managing, and removing enclosing module boundaries for the physical child cells of logical hierarchical cells. This command is typically called by the tool from the Color by Hierarchy section of the Hierarchy Exploration panel.

The command returns the created or queried module boundaries.

The enclosing module boundary can be created for specific logical hierarchical cells (*create* mode) or for all the child modules of logical hierarchical cells (*expand* mode). When creating child boundaries, the actual cells which have boundaries created is controlled by a size threshold. Cells with a size below the threshold are skipped.

The created module boundary is shaped so its rectilinear boundary encloses the bounding boxes of all the child cells in the logical hierarchy. This provides visual feedback of the standard cell and macro placement for the logical hierarchy, even if the child cells are not visible.

The created module boundary and child cells are set to the same cell color, so the association between the module boundary and child cells can be seen at all zoom levels.

The created module boundaries can also be removed (*remove* mode) or updated (*update* mode).

Note: this command extends the functionality of the existing **explore_logic_hierarchy** command to support a more complex boundary (enclosing shape) and to support consistent module boundary and child cell coloring.

NOTE

This command does not require the gui to be open.

The command supports undo i.e. the undo command will restore the module boundaries and colors to their previous state.

EXAMPLES

The following example creates an enclosing module boundary for the ALU logical hierarchical cell using the auto color.

prompt> gui_explore_logic_hierarchy -create [get_cells ALU]

The following example creates enclosing module boundaries for the child cells of the top cell using the color which is auto assigned.

prompt> gui_explore_logic_hierarchy -expand

The following example removes the module boundary for the ALU logical hierarchical.

prompt> gui_explore_logic_hierarchy -remove [get_cells ALU]

The following example will remove the enclosing module boundaries for all cells.

prompt> gui_explore_logic_hierarchy -remove

The following example updates the module boundary for the ALU logical hierarchy.

prompt> gui_explore_logic_hierarchy -update [get_cells ALU]

The following example updates the enclosing module boundaries for all cells.

prompt> gui_explore_logic_hierarchy -update

SEE ALSO

explore_logic_hierarchy(2)
set_colors(2)

gui_export_utable

Export the UserTable to a value file.

SYNTAX

string **gui_export_utable**
-name *Table_Name*
[-file *File_Name* [-tsv_mode] [-ssv_mode]]
[-filter *Filter*]
[-tsv_mode]
[-ssv_mode]
[-add_col_type]
[-add_metadata]
[-overwrite]

Table_Name String
File_Name String

ARGUMENTS

-name *Table_Name*

The name of the user table to export to a CSV value file.

-file *File_Name*

The name of the file to write the table to, by default the name of the table is used with the '.csv' extension.

-filter *Filter*

This argument allows you to filter the rows in the given table to only rows that match the filter. The filter expression is the same expression allowed in the GUI filter field of the UserTable GUI view.

-tsv_mode

This flag changes the default delimiter from the comma char to the tab char.

-ssv_mode

This flag changes the default delimiter from the comma char to the semicolon char.

-add_col_type

This flag will add column type information to be added as a postfix to the header column names. This allows type information to be available if imported again by this tool at a later time.

-add_metadata

Add metadata to the top of the exported table. See the command gui_set_utable_meta for more information.

-overwrite

This flag will force the destination file to be overwritten.

DESCRIPTION

This command exports the contents of the given UserTable into a CSV value file. The first line contains all the column names and if the **-add_col_type** flag is given, it also appends column data type information to the column names.

EXAMPLES

The following example just exports the given UserTable to a file with the same name as the table plus a '.csv' extension.

shell> **gui_export_utable -name my_table**

The following example exports the UserTable 'my_table' to the filename 'results' and uses the Tab char as a delimiter. The resultant file will be called 'results.tsv'.

shell> **gui_export_utable -name my_table -file results -tsv_mode**

SEE ALSO

gui_append_utable(2)
gui_change_selection_utable(2)
gui_close_utable(2)
gui_create_utable(2)
gui_import_utable(2)
gui_set_utable_meta(2)
gui_open_utable(2)
gui_show_utable(2)
gui_write_utable(2)

gui_filter_charts_model

Filter charts model

SYNTAX

status **gui_filter_charts_model**
-model *model_id*
-expr *string*

Data Types

model_id int

ARGUMENTS

-model *model_id*

Model to filter (previously created by **gui_create_charts_model** command).

-expr *string*

Filter expression which returns '1' for rows that should be kept in the model.

The expression can use tcl variable names that match the column name (as long as the column name is a legal tcl variable) name and can use some standard functions.

- column : get column value for row
- row : get row value for column
- cell : get cell value for row and column
- setColumn : set column value for row
- setRow : set row value for column
- setCell : set cell value for row and column
- header : get header value for column
- setHeader : set header value for column
- type : get type for column
- setType : set type for column
- map : map row number to range

- bucket : get bucket for column value
- norm : normalize column value
- rand : get random number
- rnorm : get normaized random number
- color : get color from name
- remap : map column value to range
- timeval : get time value from column with time format

DESCRIPTION

Filter data in an existing model.

EXAMPLES

TODO

SEE ALSO

gui_create_charts_plot(2)
gui_define_charts_proc(2)

gui_get_annotations

Return a collection of layout annotations

SYNTAX

string **gui_get_annotations**
[-window *window_name*]
[-group *group_id*]
[-within *region*]
[-intersect]
[-at *location*]
[-filter *filter*]
[-nocase]
[-regexp]

string *window_name*
string *group_name*
string *filter*
rectangle *region*
location *point*

ARGUMENTS

-window *window_name*

Specifies the instance name of the layout window. If not specified implies all windows.

-group *group_name*

Specifies the annotation group name. If not specified implies all groups.

-within *region*

Search for annotations contained within the specified rectangle.

-intersect

Use this option with the -within option. When -intersect is specified, all annotations that touch the search rectangle are also returned.

-at *location*

Search for annotations that are close to the specified point.

-filter *filter*

Only return annotations that satisfy the filter expression.

-nocase

Ignore case in the filter expression. Option requires -filter.

-regexp

Use regular expressions in the filter expression. Option requires -filter.

DESCRIPTION

Returns a collection of gui_annotation objects that satisfy the requirements. You can use get_attribute to query properties of the annotations. Also some attributes may be modified with set_attribute.

EXAMPLES

The following example retrieves all the annotations in the Layout.1 window.

```
shell> gui_get_annotations -window Layout.1
```

This example shows how to use the client_data attribute to find annotations with that setting.

```
shell> gui_get_annotations -filter client_data==MyData
```

This example shows how to find annotations that are within a specified rectangle

```
shell> gui_get_annotations -within {{10.5 2.0} {23.5 14.6}}
```

To also find the annotations that touch the search rectangle:

```
shell> gui_get_annotations -within {{10.5 2.0} {23.5 14.6}} -intersect
```

SEE ALSO

gui_add_annotation gui_remove_annotations

gui_get_attribute

Get the value of a GUI attribute for one or more objects.

SYNTAX

string **gui_get_attribute**
object_collection
attribute_name
[-default *default*]

string *object_collection*
string *attribute_name*
string *default*

ARGUMENTS

object_collection

A collection of one or more objects.

attribute_name

The name of a GUI attribute.

-default *default*

Default value for objects that do not support GUI attribute *attribute_name*.

DESCRIPTION

Queries the requested attribute value for each element of the object collection. The return value is a list of one or more attribute values (depending on the size of the object collection). If *default* is not specified and one of the objects in *object_collection* does not support attribute *attribute_name*, a Tcl error is raised. If *default* is specified that *default* is used for all objects that do not support attribute *attribute_name*. This command is hidden.

EXAMPLES

psyn_shell-t> **gui_get_attribute [find cell I_ALU] full_name**

I_ALU

psyn_shell-t> **gui_get_attribute [find cell U*7] full_name**
U7 U17 U27 U37

SEE ALSO

gui_set_attribute(2)

gui_get_bucket_option

Returns the value of a bucket option of a given visual or map mode.

SYNTAX

string **gui_get_bucket_option**
-map *map_name*
-bucket *bucket_name*
-option *option_name*
[-default]

Data Types

map_name	string
bucket_name	string
option_name	string

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

-bucket *bucket_name*

Specifies the name of the bucket.

-option *option_name*

Specifies the name of the option to be set.

-default

Specifies that command has to return default option value.

DESCRIPTION

This command returns an option value that are available for the bucket in the specified visual mode or map mode. You must specify the visual mode or map mode name, the bucket name, and the option name. The command can return either current or default value.

EXAMPLES

The following example returns the value of 'visible' option of 'light blue' bucket of 'Highlight' visual mode:

prompt> **gui_get_bucket_option -map {Highlight} -bucket {light blue} -option {visible}**

SEE ALSO

gui_set_bucket_option(2)
gui_get_bucket_option_list(2)
gui_get_map_list(2)
gui_get_map_option(2)
gui_get_map_option_list(2)
gui_set_map_option(2)

gui_get_bucket_option_list

Lists the available options for buckets in the specified visual mode or map mode.

SYNTAX

string **gui_get_bucket_option_list**
-map *map_name*

Data Types

map_name string

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

DESCRIPTION

This command displays a list of all options that are available for the buckets in the specified visual mode or map mode.

EXAMPLES

The following example displays a list of the available options for buckets in the global route congestion map:

prompt> **gui_get_bucket_option_list -map AREAPARTITION**

SEE ALSO

gui_get_bucket_option(2)
gui_get_map_list(2)
gui_get_map_option(2)
gui_get_map_option_list(2)

gui_set_bucket_option(2)
gui_set_map_option(2)

gui_get_category_nodes

Returns the specified category tree nodes as a collection.

SYNTAX

gui_get_category_nodes
[-tree *category_tree*]
category_nodes_list

Data Types

category_tree	string or category_tree collection
category_nodes_list	list

ARGUMENTS

-tree *category_tree*

The category tree in which the category node operation will happen; must be a category tree name or a category tree collection of size one; the most recently used category tree is used if the *-tree* option is omitted.

category_nodes_list

List of hierarchical (category node) name patterns and collections of category nodes to be returned. A category node name is a unique hierarchical name of a category node in a Tcl list format.

DESCRIPTION

Returns the specified category tree nodes as a collection.

You should use this command only if at least one category tree is available. The command acts on the category tree specified.

EXAMPLES

The following example shows how to get the ALL root category node as a collection.

```
shell> set my_clct [gui_get_category_nodes {ALL}]
```

The following example shows how to return (as a collection) the {Launch Clock: my_clock} category node that is a child of the {Pathgroup: my_clock} category node, which in turn is a child of the ALL root category node.

```
shell> set my_clct [gui_get_category_nodes \
? {{ALL {Pathgroup: my_clock} {Launch Clock: my_clock}}}]
```

SEE ALSO

gui_create_category_nodes(2)
gui_remove_category_nodes(2)

gui_get_category_trees

Returns the specified category trees (or all category trees by default) in a collection.

SYNTAX

category_tree_collection **gui_get_category_trees**
[*category_trees_list*]

Data Types

category_tree_collection collection (of category tree objects)
category_trees_list list (of category tree name patterns and collections)

ARGUMENTS

category_trees_list

A list of of category tree name patterns and category tree collections specifying the category trees to be returned in the result collection. If this option is omitted, then all existing category trees are returned in the result collection.

Each category tree name pattern must be either the exact name of an existing tree, or the pattern "*", which means "all category trees". More general forms of wild-card matching are not supported at this time.

DESCRIPTION

Returns the specified category trees (or all category trees by default) in a category tree collection. An empty string is returned if no matching category trees exist.

EXAMPLES

This example show the creation of a couple of category trees, and the results of several different runs of gui_get_category_trees:

```
shell> set path_clct [get_timing_paths ...]
_sel18
shell> set path_clct2 [get_timing_paths ...]
_sel19
shell> gui_create_category_tree -name tree0 -collections $path_clct
```

```
_sel20
shell> gui_create_category_tree -name tree1 -collections $path_clct2
_sel21
shell> get_attribute [gui_get_category_trees "tree1 tree0"] name
tree1 tree0
shell> get_attribute [gui_get_category_trees tree1] name
tree1
shell> get_attribute [gui_get_category_trees *] name
tree0 tree1
```

SEE ALSO

gui_create_category_tree(2)
gui_remove_category_trees(2)
get_attribute(2)

gui_get_cell_block_marks

Gets a list of the block mark string values on one or more cells.

SYNTAX

list **gui_get_cell_block_marks**
cells

Data Types

cells list

ARGUMENTS

cells

Lists one or more cell name patterns or cell collections.

DESCRIPTION

This command gets the block mark string values from one or more hierarchical or leaf-level cell instances.

EXAMPLES

The following example sets the block mark for a hierarchical cell instance identified by a cell name, and then gets the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU  ALU
prompt> gui_get_cell_block_marks I_TOP/I_ALU
ALU
```

The following example sets the block mark for a hierarchical cell instance identified by a nested run of the **get_cells** command, and then gets the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks [get_cells I_TOP/I_ALU]  ALU
prompt> gui_get_cell_block_marks [get_cells I_TOP/I_ALU]
ALU
```

The following example shows how using the **get_attribute** command to query the **block_mark** attribute is an alternative to using the **gui_get_cell_block_marks** command:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU  ALU
prompt> gui_get_cell_block_marks I_TOP/I_ALU
ALU
prompt> get_attribute -class cell I_TOP/I_ALU block_mark
ALU
```

SEE ALSO

gui_set_cell_block_marks(2)
gui_remove_cell_block_marks(2)

gui_get_charts_data

Get charts data

SYNTAX

value **gui_get_charts_data**
{ -global | -model *model_id* | -view *view_name* |
-plot *plot_name* | -annotation *annotation_name* }
[-object *object_id*]
[-column *column_name*]
[{ -header | -row *int* }]
-name *string*
[-data *string*]

Data Types

model_id int
view_name string
plot_name string
annotation_name string
object_id string
column_name int
value string

ARGUMENTS

-global

Get global data.

-model *model_id*

Model to get data from.

-view *view_name*

View to get data from.

-plot *plot_name*

Plot to get data from.

-annotation *annotation_name*

Annotation to get data from.

-object *object_id*

Plot object to get data from.

Only used by the *plot* option.

-column *column_name*

Column to get model data from.

Only used by the *model* option.

-header

get model data from header.

Only used by the *model* option.

-row *int*

Row to get model data from.

Only used by the *model* option.

-name *string*

Name of data to return.

The supported names depend on which object the data is extracted from:

For model the following names are supported:

value Get value of model horizontal header (using the *header* option) value or row value (using the *row* option). The column is specified using the *column* option.

num_rows Get number of rows in model.

num_columns Get number of columns in model.

hierarchical Is model hierarchical.

header Get model header names. If column is specified using the *column* option then only the header for that column is returned.

row Get model row values for the row specified using *row* option.

column Get model column values for the column specified using *column* option.

duplicates Get rows with duplicate values (same value in all columns).

column_index Get model column index from name specified in *data* option.

title Get the model title.

The following data names are queried on the specifed column (single value) or, if the column is not specified, then on all columns (list of values):

type Get model column type.

min or minimum Get model minimum column value.

max or maximum Get model maximum column value.

mean or avg or average Get model average column value.

monotonic Get if model column monotonic (increasing or descreasing)

increasing Get if model column increasing

num_unique Get number of unique model values in column.

unique_values Get unique model values in column.

unique_counts Get counts of each unique model value in column.

num_null Get number of null values in column.

median Get model median column value.

lower_median Get model lower median column value.

upper_median Get model upper median column value.

stddev or std_dev Get model standard deviation value.

outliers Get model outlier values.

For view the following names are supported:

plots return a list of all plots in the view.

annotations return a list of all annotations in the view.

selected_objects return a list of all selected objects in the view.

For plot the following names are supported:

model Model index.

view Parent view name.

annotations Get plot's annotation object ids.

objects Get plot's object ids.

selected_objects Get plot's selected object ids.

For annotation the following names are supported:

view Parent view name.

plot Parent plot name.

The following global data the following names are supported:

models Get model indices

views Get view names.

plot_types Get plot type names.

plots Get all plots names.

column_types Get all column type names.

column_type.names Get names of parameters for specified column type (in *data* option).

annotation_types Get all annotation type names.

symbol_names Get all symbol names.

procs Get all user defined procedure names.

You can also use the special name "?" which will return a list of the supported names for the specified object.

-data ***string***

Extra data needed to query some data values.

RETURNS

value

Returned value.

DESCRIPTION

Get data from model, view, plot type, plot objects or from global object.

If model is specified then the *column* and *header* or *row* should be specified.

EXAMPLES

The following example gets all the plots of a view.

shell> gui_get_charts_data -view $view -name plots

SEE ALSO

gui_set_charts_data(2)

gui_get_charts_property

Get charts property

SYNTAX

value **gui_get_charts_property**
{ -view *view_name* | -plot *plot_name* |
-annotation *annotation_id* }
[-object *object_id*]
-name *string*

Data Types

view_name string
plot_name string
annotation_id string
object_id string
value string

ARGUMENTS

-view *view_name*

View to get property from.

-plot *plot_name*

Plot to get property from.

-annotation *annotation_id*

Plot or view annotation to get property from.

-object *object_id*

Plot object to get property from.

The *plot* option must be specified as well.

-name *string*

Name of property to return.

The supported names depend on which object the property is extracted from but can be listed by using the special property name "?".

RETURNS

value

Returned value.

DESCRIPTION

Get property from view, plot, view annotation, plot annotation or plot objects.

EXAMPLES

The following example gets all the position of a annotation in a plot.

```
shell> gui_get_charts_property -plot $plot -annotation $annotation -name position
```

SEE ALSO

gui_set_charts_property(2)

gui_get_clock_tree

Returns clock tree fanin or fanout cone for given clock and start pin.

SYNTAX

status **gui_get_clock_tree**
-clocks *clocks*
-from *from_pins*
-trace_direction forward | backward | both

Data Types

clocks string or collection
from_pins string or collection

ARGUMENTS

-clocks *clocks*

Specifies the clocks on which the clock tree fanin or fanout cone is calculated.

-from *from_pins*

Specifies the start pins from which the fanin or fanout cone is traced.

-trace_direction forward | backward | both

Specifies the trace direction. The trace direction must be one of following values:

- forward: Trace forward through the clock tree (get fanout pins)
- backward: Trace backward through the clock tree (get fanin pins)
- both: Trace both forward and backward (get both fanout and fanin pins)

DESCRIPTION

Returns a collection of clock tree fanin or fanout pins for given clock and start pin.

EXAMPLES

The following example returns the fanout pins of clock CLK3, starting from selected pin.

```
prompt> gui_get_clock_tree \
  -clocks [get_clocks CLK3] -from [get_selection] -trace_direction forward
```

SEE ALSO

get_clocks(2)

gui_get_color_value

Get the RGB color value for given color index or name, else return all color information for the palette.

SYNTAX

string **gui_get_color_value** *[index]*
[-palette string]
[-color_name string]

ARGUMENTS

index

Use given color index number. An error message is posted if the given index is greater than the maximum index allowed for the current or given Palette. This argument is exclusive with option -color_name.

-palette

User given palette name, the default is 'highcontrast'. Valid values are: highcontrast, highlight, qt, style. An error message is printed if an invalid palette name is given.

-color_name

Use given color name. Must be valid for the current/given palette. If an invalid color name is given an error message is posted. This option is exclusive with the 'index' argument.

DESCRIPTION

Get the RGB color value string for the given color name or index value.

If no argument is given the command will return a TCL list of lists. One list for every color in the current/given palette. The list for each color contains the index number, the RGB color value and, if available, a color name.

EXAMPLES

shell> **gui_get_color_value 10**
#005000

```
shell> gui_get_color_value -color_name magenta
#ff00ff

shell> gui_get_color_value
{0 #000000 }
{1 #464646 }
{2 #00ff64 }
{3 #78bec8 }
{4 #ff00ff magenta}
{5 #00af96 }
{6 #0000be }
{7 #00ffff cyan}
{8 #b1f476 light_green}
{...}
```

gui_get_current_category_tree

Returns the "current category tree" (if any) in a collection of size 1.

SYNTAX

category_tree_collection **gui_get_current_category_tree**

Data Types

category_tree_collection collection (of one category tree object)

DESCRIPTION

Returns the "current category tree" (if any) in a collection of size 1. An empty string is returned if there are no existing category trees.

At any time, exactly one of the existing category trees has the status "current category tree". When a new category tree is created, it immediately becomes the current category tree. If the current category tree is removed, a different category tree will be chosen as the current category tree. If there are no existing category trees, then there is no current category tree.

EXAMPLES

Here are examples of getting and setting the current category tree:

```
shell> get_attribute [gui_get_category_trees *] name
_tree008 _tree009 _tree010 _tree011
shell> get_attribute [gui_get_current_category_tree] name
_tree009
shell> gui_set_current_category_tree _tree008
shell> get_attribute [gui_get_current_category_tree] name
_tree008
```

SEE ALSO

gui_create_category_tree(2)
gui_remove_category_trees(2)
gui_get_category_trees(2)

gui_set_current_category_tree(2)
get_attribute(2)

gui_get_current_task

Get the name of the current task.

SYNTAX

string **gui_get_current_task**

DESCRIPTION

The gui_get_current_task command queries the name of the current task.

SEE ALSO

gui_create_task(2)
gui_get_current_task_item(2)
gui_get_current_task_page(2)
gui_set_current_task(2)
gui_get_task_list(2)

gui_get_current_task_item

Get the name of the current task item.

SYNTAX

string **gui_get_current_task_item**

DESCRIPTION

The gui_get_current_task_item command queries the name of the current task item.

SEE ALSO

gui_get_current_task(2)
gui_get_current_task_page(2)
gui_set_current_task(2)
gui_get_task_list(2)

gui_get_current_task_page

Get the name of the current task page.

SYNTAX

string **gui_get_current_task page**

DESCRIPTION

The gui_get_current_task_page command queries the name of the current task page.

SEE ALSO

gui_create_task(2)
gui_get_current_task(2)
gui_get_current_task_item(2)
gui_set_current_task(2)
gui_get_task_list(2)

gui_get_current_window

Retrieves the window ID of the active top-level or view window.

SYNTAX

string **gui_get_current_window**
[-toplevel | -view | -parent *parent_window_id*]
[-hier_name]
[-types *window_type_list*]
[-mru]

Data Types

parent_window_id string
window_type_list list

ARGUMENTS

-toplevel

Retrieves the active top-level window unless you also specify the **-types** option.

If you also specify the **-types** option, the **-toplevel** option retrieves the most recently active top-level window that has one of the specified window types.

The **-toplevel**, **-view**, and **-parent** options are all mutually exclusive.

-view

Retrieves the active view window in the active top-level window unless you also specify the **-types** option, the **-mru** option, or both.

if you also specify the **-types** option but not the **-mru** option, the **-view** option retrieves the most recently active view window with one of the specified window types in the active top-level window.

if you also specify the **-mru** option but not the **-types** option, the **-view** option retrieves the most recently active view window in any of the open top-level windows.

If you also specify both the **-types** option and the **-mru** option, the **-view** option retrieves the most recently active view window that has one of the specified window types and is in an open top-level window.

The **-view**, **-toplevel**, and **-parent** options are all mutually exclusive.

-parent *parent_window_id*

Retrieves the active view window in the specified top-level window unless you also specify the **-types** option.

If you also specify the **-types** option, the **-parent** option retrieves the most recently active view window that has any of the specified window types and is in the specified top-level window.

The **-parent**, **-toplevel**, and **-view** options are all mutually exclusive.

-hier_name

Returns the window ID in Qtcl format, which is similar to a full path name for the window, for use with the qtcl_* commands.

Note that a value returned in this format is not usable by gui_* commands.

-types *window_type_list*

Forces the tool to retrieve a window with one of the specified window types.

Use this option to restrict the search to certain types of windows. If you specify multiple types, they should be consistent (all toplevel window types or all view window types), but nonconsistency is not considered an error. The first type in the list determines whether the tool retrieves a top-level window or a view window when the **-toplevel** and **-view** options are not specified.

When you use the **-types** option, you should not use the **-toplevel** option or the **-view** option because the tool can determine from the list whether the window should be a top-level window or a view window. If you specify one of these options, any types that do not correspond to the option are silently ignored. For example, if you specify **-view**, then all top-level types are ignored.

-mru

Forces the tool to retrieve the most recently active view window from among all the open top-level and view windows. This option is effective only when you specify the **-view** option or when all the window types you specify with the **-types** option are view window types.

DESCRIPTION

This command retrieves the window ID of the active top-level or view window based on the options that you specify, or returns an empty string if no match is found.

A view window is a child window of a top-level window and is created with the **gui_create_window** command.

If you do not specify the **-toplevel**, **-view**, and **-parent** options, the tool uses the **-types** option to determine the window type. If you also do not specify the **-types** option, the tool uses the **-toplevel** option by default.

EXAMPLES

The following example retrieves the active top-level window:

```
prompt> gui_get_current_window -toplevel
```

The following example retrieves the active view window:

```
prompt> gui_get_current_window -view
```

The following example retrieves the most recently active layout view:

```
prompt> gui_get_current_window -types "Layout" -mru
```

The following example return $cons:

```
prompt> set top1 [gui_create_window -type LayoutWindow]
prompt> set top2 [gui_create_window -type LayoutWindow]
prompt> set cons [gui_create_window -type Layout -parent $top]
prompt> gui_get_current_window -parent $top
```

SEE ALSO

gui_close_window(2)
gui_exist_window(2)
gui_show_window(2)
gui_get_window_types(2)
gui_get_window_ids(2)
gui_set_active_window(2)
gui_create_window(2)

gui_get_display_view

Returns the display view type (abstract, design, frame, layout, or outline) used in the layout when cell-specific views are active. Views are set with the **gui_set_display_view** command.

SYNTAX

string **gui_get_display_view**
-cells *cell_list*

Data Types

cell_list list

ARGUMENTS

-cells *cell_list*

Specifies the physical cells for which to report the view.

DESCRIPTION

This command retrieves the cell-specific display view value. For this command, the view is set with the **gui_set_display_view** manpage for more detailed information.

EXAMPLES

The following example returns the view setting for cells A and B. The first command shows that cell A has no display view set. The second command shows that the display view type for cell is design.

```
prompt> gui_get_display_view -cells A
prompt> gui_get_display_view -cells B
design
```

SEE ALSO

gui_set_display_view(2)

gui_get_error_browser_option

Get Error Browser Dialog option values

SYNTAX

string **gui_get_error_browser_option**
-grouping
| -show_mode
| -view_mode
| -zoom_factor
| -hide_fixed
| -hide_ignored
| -advance_to_next_unfixed_error
| -dim
| -show_open_locator_nodes
| -show_tooltip
| -show_command_buttons
| -highlight_objects
| -auto_set_object_visible
| -auto_set_object_visible_type

ARGUMENTS

-grouping | -show_mode | -view_mode | -zoom_factor | -hide_fixed | -hide_ignored | -advance_to_next_unfixed_error | -dim | -show_open_locator_nodes | -show_tooltip -show_command_buttons | -highlight_objects | -auto_set_object_visible | -auto_set_object_visible_type

Mutually exclusive required option to specify the Error Browser option value to query.

DESCRIPTION

This command gets the value of the requested option setting for the Error Browser. One option value can be queried at a time.

The **-grouping** option returns *type* or *layer* which indicates whether errors are grouped by types or layers.

The **-show_mode** option returns *all*, *selected*, or *none* which indicates whether all errors, selected errors, are no errors are shown in layout views.

The **-view_mode** option returns *zoom*, *pan*, of *off* which indicates whether the layout view zooms to, pans to, or does not change to

the currently selected errors in the Error Browser.

The **-zoom_factor** option returns a float value between *0.1* and *10.0* and indicates the zoom factor used when the view_mode is zoom.

The **-hide_fixed** option returns *true* to mean "hidden" or *false* to mean "not hidden" and indicates whether fixed errors are hidden from display in the Error Browser and layout views.

The **-hide_ignored** option returns *true* to mean "hidden" or *false* to mean "not hidden" and indicates whether ignored errors are hidden from display in the Error Browser and layout views.

The **-advance_to_next_unfixed** option returns *true* to mean advance to next unfixed error when an error is fixed or *false* to mean advanced to next fixed/unfixed error.

The **-dim** option returns *true* to mean "dimmed" or *false* to mean "not dimmed" and indicates whether layout views are dimmed when errors are displayed.

The **-show_open_locator_nodes** returns *true* to mean "highlighted" or *false* to mean "not highlighted" and indicates whether net nodes are highlighted along with the open locator flylines for selected open locator errors.

The **-show_tooltip** returns *true* to mean show tooltips or *false* to mean don't show tooltips in the error browser list and details panes.

The **-show_command_buttons** returns *true* to mean show the commannd buttons or *false* to mean don't show the command buttons in the error browser.

The **-highlight_objects** returns *true* to mean "highlighted" or *false* to mean "not highlighted" and indicates whether associated design objects are highlighted in the layout view for selected errors.

The **-auto_set_object_visible** returns whether auto set layout/object visibility is turned on or not.

The **-auto_set_object_visible_type** returns whether auto set layout/object visibility turns on layers/objects incrementally (the default) or exclusively.

EXAMPLES

The following example gets the grouping option

```
gui_get_error_browser_option -grouping
```

The following example gets whether layout views are dimmed when errors are displayed

```
gui_get_error_browser_option -dim
```

SEE ALSO

gui_error_browser(2)
gui_set_error_browser_option(2)

gui_get_error_data

Creates a collection of physical DRC error data that are currently shown in the error browser.

SYNTAX

collection **gui_get_error_data**

DESCRIPTION

The **gui_get_error_data** command creates a collection of physical DRC error data that are currently shown in the error browser.

You can use the **get_drc_error_data** command at the command prompt, or you can nest it as an argument to another command (for example, **query_objects**). In addition, you can assign the **gui_get_error_data** result to a variable.

When issued from the command prompt, **gui_get_error_data** behaves as though **query_objects** had been called to display the objects in the collection.

EXAMPLES

The following example queries the physical DRC error data that are currently shown in the error browser. Although the output looks like a list, it is not. The output is just a display.

```
prompt> gui_get_drc_error_data
{"my_design_dppinassgn.err"}
```

SEE ALSO

gui_open_error_data(2)
gui_close_error_data(2)

gui_get_hierview_data

Get various state about the Hierarchy View and its sub-views. [Default return last used Hierarchy View name]

SYNTAX

string **gui_get_hierview_data**
[-view *View_Name*]
[-tree_agroup]
[-tree_filter]
[-map_area]
[-map_color]
[-childlist_name]
[-childlist_agroup]
[-childlist_filter]
[-sub_view]
[-columns]
[-values *Column_Name*]

View_Name String
Column_Name String

ARGUMENTS

-view *View_Name*

Use the given HierView window name (ex: Hier.1). Default is to get the most recent used HierView window name.

-tree_agroup

Get and return the current Hier Tree attribute group name.

-tree_filter

Get and return the current Hier Tree filter expression.

-map_area

Get and return the Hier Map Area attribute group name.

-map_color

Get and return the Hier Map Color attribute group name.

-childlist_name

Get and return the current ChildList sub-view name.

-childlist_agroup

Get and return the current ChildList attribute group name.

-childlist_filter

Get and return the current ChildList filter expression.

-sub_view

Get and return the current active sub-view type. The sub-view types are 'hier_tree', 'hier_map' and 'child_list'.

-columns

Get and return a list of all column name(s) of all selected table cells.

-values *Column_Name*

Get and return a list of all selected table cell value(s) in the given column name. **Note**: Any of the column value(s) in the row of the selected cell(s) can be retrieved.

DESCRIPTION

This command allows you to get various current Hierarchy View related view settings
and selected table cell data.

SEE ALSO

gui_set_hierview_data(2)

gui_get_highlight

Get a collection of highlighted objects.

SYNTAX

string **gui_get_highlight**

[-color *color_id* | -all_colors]
[-return_select_bus | -more_than]

string *color_id*
collection *clct*

ARGUMENTS

-color *color_id*

Specifies the color of objects to be returned. If neither -color nor -all_colors is used, the current color is assumed. The allowed colors are yellow, orange, red, green, blue, purple, light_orange, light_red, light_green, light_blue, and light_purple.

-all_colors

Specifies that all highlighted objects are to be returned.

-return_select_bus

Create a new selection bus in which to store the result instead of returning a collection of objects.

-more_than

Return 1 if more than the specified number of objects would be returned, otherwise 0.

DESCRIPTION

The gui_get_highlight command returns a collection of objects highlighted with the specified colors.

EXAMPLES

Get a collection of green highlighted objects.

```
shell> gui_get_highlight -color green
```

Get a collection of all highlighted objects.

```
shell> gui_get_highlight -all_colors
```

Query if any objects are highlighted

```
shell> gui_get_highlight -all_colors -more_than 0
```

SEE ALSO

gui_change_highlight(2)
gui_get_highlight_options(2)
gui_set_highlight_options(2)

gui_get_highlight_options

Query the options that control highlighting.

SYNTAX

string **gui_get_highlight_options**

[-current_color | -all_colors | -auto_cycle_color]

ARGUMENTS

-current_color

This option returns a string which is the current highlight color, or the default color for highlighting operations.

-all_colors

This option returns Tcl list of all of the colors which are allowed to be used for highlighting.

-auto_cycle_color

This option returns the current value of the auto cycle setting ("true" or "false").

DESCRIPTION

The gui_get_highlight_options command queries the settings which control highlighting. Exactly one option must be provided.

EXAMPLES

Get the current highlight color.

```
shell> gui_get_highlight_options -current_color
```

Get all highlight colors.

```
shell> gui_get_highlight_options -all_colors
```

SEE ALSO

gui_change_highlight(2)
gui_get_highlight(2)
gui_set_highlight_options(2)

gui_get_layer_widths

Returns a list of widths per layer for route editing.

SYNTAX

values **gui_get_layer_widths**
-layer *layer_name*
[{ -default | -current | -user }]

Data Types

layer_name string
values list

ARGUMENTS

-layer *layer_name*

Specifies the routing layer for which to query the widths.

-default

Returns the list of widths to use by default, if the user has not set up their own list of layers.

The default list consists of:

- layer minimum width
- layer default width
- any NDR widths for layer

-current

Returns the list of widths that is used by the editor. If the user specified a list then it will be returned, otherwise the default list will be returned.

-user

Returns the list of widths explicitly specified by the user. This may be an empty list if no values were specified by the user via **gui_set_layer_widths**.

DESCRIPTION

This command returns a list of discrete widths; these widths are used by the route editor.

By default the value returned by *-current* will be returned.

EXAMPLES

The following example returns the user defined widths for layer 'M1'.

prompt> **gui_get_layer_widths -layer M1 -user**

SEE ALSO

gui_set_layer_widths(2)

gui_get_map

Retrieves attributes for a specified map mode.

SYNTAX

string **gui_get_map**
-name *identifier*
[-buckets]
[-title]
[-help_topic]
[-infotip]
[-discrete]
[-icon_file]
[-update_cmd]
[-float]
[-top_exaggeration]
[-mid_exaggeration]
[-bot_exaggeration]

Data Types

identifier string

ARGUMENTS

-name *identifier*

Specifies the name of the map mode to be queried.

-buckets

Returns a list of bucket names. The bucket names are retrieved in rendering order, starting from the first to the last bucket rendered.

-title

Returns the title string.

-help_topic

Returns the help topic string.

-infotip

Returns the InfoTip string.

-discrete

Returns a true or false value that indicates whether the buckets of this map mode are discrete or continuous. Only discrete buckets can be reordered. This attribute is set when this map mode was created, and cannot be changed.

-icon_file

Returns the gui menu icon file.

-update_cmd

Returns the Tcl command that is executed when the map mode is accessed. A typical use for this option is to update the state of the map mode if its contents are likely to change under certain circumstances.

-float

Returns a true or false value indicating whether the map mode buckets have a floating point range.

-top_exaggeration

Returns the top bucket exaggeration.

-mid_exaggeration

Returns the middle bucket exaggeration.

-bot_exaggeration

Returns the bottom bucket exaggeration.

DESCRIPTION

The **gui_get_map** command queries the option values of an existing map mode.

EXAMPLES

To retrieve names of the buckets in the map mode named "densityMap", use the following command:

prompt>**gui_get_map -name densityMap -buckets**

SEE ALSO

gui_get_mapbucket(2)

gui_get_map_list

Lists the current visual and map modes.

SYNTAX

string **gui_get_map_list**

ARGUMENTS

The **gui_get_map_list** command has no arguments.

DESCRIPTION

The **gui_get_map_list** command returns a list of existing visual and map modes.

EXAMPLES

The following example gets the names of all available visual and map modes:

```
prompt> gui_get_map_list
```

SEE ALSO

gui_create_vm(2)
gui_create_vmbucket(2)
gui_get_bucket_option(2)
gui_get_bucket_option_list(2)
gui_get_map_option(2)
gui_get_map_option_list(2)
gui_get_vm(2)
gui_get_vmbucket(2)
gui_remove_vm(2)
gui_remove_vmbucket(2)

gui_set_bucket_option(2)
gui_set_map_option(2)
gui_set_vm(2)
gui_set_vmbucket(2)
gui_show_map(2)
gui_update_vm(2)

gui_get_map_option

Gets the value for an option in the specified visual mode or map mode.

SYNTAX

string **gui_get_map_option**
-map *map_name*
-option *option_name*
[-default]

Data Types

map_name string
option_name string

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

-option *option_name*

Specifies the name of the option to be set.

-default

Gets the option default value.

DESCRIPTION

This command gets the value for an option in a visual mode or map mode. You must specify the visual mode or map mode name and the option name. You can get set option specific value or its default value.

EXAMPLES

The following example gets the number of bins in the global route congestion map:

```
prompt> gui_get_map_option -map AREAPARTITION \
    -option num_bins -value 9
```

SEE ALSO

gui_get_bucket_option(2)
gui_get_bucket_option_list(2)
gui_get_map_list(2)
gui_set_map_option(2)
gui_get_map_option_list(2)
gui_set_bucket_option(2)

gui_get_map_option_list

Lists the available options for the specified visual mode or map mode.

SYNTAX

string **gui_get_map_option_list**
-map *map_name*

Data Types

map_name string

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

DESCRIPTION

This command displays a list of all options that are available for the specified visual mode or map mode.

EXAMPLES

The following example displays a list of the available options for the global route congestion map:

prompt> **gui_get_map_option_list -map AREAPARTITION**

SEE ALSO

gui_get_bucket_option(2)
gui_get_bucket_option_list(2)
gui_get_map_list(2)
gui_get_map_option(2)

gui_set_bucket_option(2)
gui_set_map_option(2)

gui_get_mapbucket

Gets attributes for Map Mode Bucket

SYNTAX

string **gui_get_mapbucket**
-map *mode_identifier*
-name *bucket_identifier*
[-infotip]
[-color]
[-pattern]
[-exaggeration]
[-number]
[-maxval]
[-minval]
[-visible]
[-special]
[-title]
[-objcount]

Data Types

mode_identifier string
bucket_identifier string

ARGUMENTS

-map *mode_identifier*

Specifies the map mode to which the bucket is a member. The map mode must already exist.

-name *bucket_identifier*

Specifies the name of the map mode bucket to be queried. To see the list of buckets associated with a specific map mode use the gui_get_map command with the -buckets option.

-infotip

Return infotip string.

-color

Return bucket rendering color.

-pattern

Return bucket rendering fill pattern.

-exaggeration

Return bucket min pixel exaggeration value.

-number

Return bucket display number.

-maxval

Return bucket maximum value.

-minval

Return bucket minimum value.

-visible

Return visibility.

-special

Return whether bucket is special.

-title

Return bucket title.

-objcount

Return the number of objects in the bucket.

DESCRIPTION

The gui_get_mapbucket command queries an instance of a map mode bucket for the specified option values. The values are returned in a list of option-value pairs.

EXAMPLES

Query the current color for the bucket named "Bucket10" in the map mode "densityMap":

prompt> **gui_get_mapbucket -map densityMap -name Bucket10 -color**

Query the current color and pattern for the bucket named "Bucket10" in the map mode "densityMap":

prompt> **gui_get_mapbucket -map densityMap -name Bucket10 -color -pattern**

SEE ALSO

gui_get_map(2)

gui_get_menu_roots

Return a sorted list of all menu roots used by the application.

DESCRIPTION

This command returns a list of all the menu roots used by the application. The menu roots include menu roots used by toplevel menu bars and menu roots used by context menus. A menu root is used to group menu items that share the same menu root together. All items under each toplevel window's menu bar shares the same menu root; similarly, all items in a context menu (menu brought up with right mouse click in a window) shares the same context menu root.

SEE ALSO

gui_create_menu(2)

gui_get_mouse_tool_option

Get the value of a mouse tool option

SYNTAX

string **gui_get_mouse_tool_option** -tool *string*
-option *string*

string *string*
string *string*

ARGUMENTS

-tool *string*

Tool to get option for.

-option *string*

Option to get.

DESCRIPTION

This command returns value of a mouse tool option .

EXAMPLES

> gui_get_mouse_tool_option -tool SELECT -option InputMode
Smart

SEE ALSO

gui_mouse_tool(2)
gui_set_mouse_tool_option(2)

gui_get_performance_log_option

Gets the current option value controlling performance log behavior.

SYNTAX

string **gui_get_performance_log_option**

Data Types

ARGUMENTS

DESCRIPTION

This command is used to query the current setting for the performance log which is set by gui_set_performance_log_option.

EXAMPLES

The following example sets, then gets the current setting.

prompt> **gui_set_performance_log_option -commands {gui_zoom win_select_objects}**
prompt> **gui_get_performance_log_option**
commands: gui_zoom win_select_objects

SEE ALSO

gui_log_performance(2)
gui_set_performance_log_option(2)

gui_get_pref_categories

Return a list of the current preference categories.

SYNTAX

string **gui_get_pref_categories**

DESCRIPTION

This command returns a tcl list of the current preference categories.

SEE ALSO

gui_create_pref_category(2)
gui_create_pref_key(2)
gui_set_pref_value(2)
gui_get_pref_value(2)
gui_get_pref_value_type(2)
gui_remove_pref_key(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_get_pref_keys

Return a list of preference keys under the specified category.

SYNTAX

string **gui_get_pref_keys** -category *Category*

Category *String*

ARGUMENTS

-category *Category*

Category specifies the category to use.

DESCRIPTION

Return a list of preference keys under the specified category.

SEE ALSO

gui_create_pref_category(2)
gui_create_pref_key(2)
gui_set_pref_value(2)
gui_get_pref_value(2)
gui_get_pref_value_type(2)
gui_remove_pref_key(2)
gui_get_pref_categories(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_get_pref_value

Get the value of a preference key.

SYNTAX

string **gui_get_pref_value** -key *Key* [-category *Category*]

Key *String*
Category *String*

ARGUMENTS

-key *Key*

Key specifies the key to retrieve the value from.

-category *Category*

Category specifies the category to search for the key. If not specified, a default category will be used.

DESCRIPTION

This command retrieves the value of a preference key from the specified key name and category.

EXAMPLES

The following example create a user-defined category *some_category* for storing preferences and then create a boolean key *some_key* under that category with the initial value of false. The command **gui_get_pref_value** is then used to retrieve the value of the preference key just created.

```
gui_create_pref_category -category some_category
gui_create_pref_key -category my_category -key some_key -value_type bool -value false
set x [gui_get_pref_value -category my_category -key some_key]
```

SEE ALSO

gui_create_pref_category(2)
gui_create_pref_key(2)
gui_set_pref_value(2)
gui_get_pref_value_type(2)
gui_remove_pref_key(2)
gui_get_pref_categories(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_get_pref_value_type

Get the data type of the value of a preference key.

SYNTAX

string **gui_get_pref_value_type** -key *Key* [-category *Category*]

Key *String*
Category *String*

ARGUMENTS

-key *Key*

Key specifies the key to retrieve the value type from.

-category *Category*

Category specifies the category to search for the key. If not specified, a default category will be used for searching the key.

DESCRIPTION

This command retrieves the data type of the value of a preference key from the specified key name and category.

EXAMPLES

The following example create a user-defined category *some_category* for storing preferences and then create an integer key *some_key* under that category with the initial value of 200. The command **gui_get_pref_value_type** is then used to retrieve the data type of the value of the preference key just created.

```
gui_create_pref_category -category some_category
gui_create_pref_key -category my_category -key some_key -value_type integer -value 200
set x [gui_get_pref_value_type -category my_category -key some_key]
{comment: x should be equal to "integer" }
```

SEE ALSO

gui_create_pref_category(2)
gui_create_pref_key(2)
gui_set_pref_value(2)
gui_get_pref_value_type_type(2)
gui_remove_pref_key(2)
gui_get_pref_categories(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_get_presets

Gets list of preset name for object specified by category

SYNTAX

string **gui_get_presets**
  -category *category*
  [-system]
  [-shared]

Data Types

category string

ARGUMENTS

-category *category*

Specifies the name of the category that the presets be retrieved from.

-system

Return system presets. This option is mutually exclusive with the -shared option.

-shared

Return shared presets. This option is mutually exclusive with the -system option.

DESCRIPTION

This command retrieves the list of presets for a category.

EXAMPLES

The following example gets the system presets for the Layout.

prompt> **gui_get_presets -category Layout -system**

SEE ALSO

gui_set_preset(2)

gui_get_region

Returns the coordinates of the current region rectangle or rectilinear polygon.

SYNTAX

list **gui_get_region**

DESCRIPTION

This command returns the coordinates of the rectangle or rectilinear polygon that defines the current region in layout region mode.

Note that the preferred command for typical extension writing is usually **gui_set_layout_user_command** instead of **gui_get_region** because **gui_set_layout_user_command** has snapping, mouse up drag, cancel, and user prompt capabilities.

EXAMPLES

```
prompt> gui_get_region
{{-1743.945 18.357} {42.833 1523.657}}
```

SEE ALSO

gui_set_region(2)
gui_set_layout_user_command(2)

gui_get_setting

Gets a setting on the specified window.

SYNTAX

string **gui_get_setting**
  -window *WindowID*
  { -setting *Setting* | -list }

WindowID *String*
Setting *String*

ARGUMENTS

-window *WindowID*

WindowID specifies the window to get the setting

-setting *Setting*

Setting specifies the setting to get

-list

Specifies that the list of available settings needs to be returned

DESCRIPTION

Gets the list of available settings on the specified window. Windows are views or top level windows. The setting is a property of the window. The returned value is a string which has the type of the setting being read. Some windows might not have this function implemented.

EXAMPLES

shell> **gui_get_setting -window Layout.1 -setting showCell**
1

```
shell> gui_get_setting -window Layout.1 -setting viewLevel
2
```

SEE ALSO

gui_set_setting(2)

gui_get_task_list

Lists all the available task names.

SYNTAX

string **gui_get_task_list**

DESCRIPTION

The gui_get_task_list command queries the names of all available tasks.

DESCRIPTION

prompt> **gui_get_task_list** all {Block Implementation} {Design Planning}

SEE ALSO

gui_create_task(2)
gui_remove_task(2)
gui_set_task_list(2)
gui_set_current_task(2)
gui_get_current_task(2)

gui_get_task_page

Return the name of the task page for the given task item.

SYNTAX

string **gui_get_task_page**

SYNTAX

gui_get_task_page
-task *TaskName*
-item *ItemName*

TaskName *String*
ItemName *String*

ARGUMENTS

-task *TaskName*

TaskName specifies the name of the task.

-item *ItemName*

ItemName specifies the hierarchical name of the task item for which to get the page name",

DESCRIPTION

This command returns the name of the task page for the task item identified by the option values. This is useful when you want to create a new task item re-using a task page that is invoked by an existing task item.

EXAMPLES

To reuse the task page from the built-in task item "Pin Assignment->Pin Placement" in the task "Hierarchical Design" in your own task flow, you can do the following:

```
prompt> set pageName [gui_get_task_page -task "Hierarchical Design"
                -item "Pin Assignment->Pin Placement"]
prompt> gui_create_task_item -task myTask -name myItemName
                -page $pageName
```

SEE ALSO

gui_create_task_item(2)
gui_set_current_task(2)
gui_get_task_list(2)

gui_get_toolbar_names

Return a Tcl list of the names of all the toolbars that have been created with the **gui_create_toolbar** command.

SYNTAX

void **gui_get_toolbar_names**
[-window *WindowId*]

WindowId *String*

ARGUMENTS

-window *WindowId*

Return toolbars defined in this window. If omitted, return toolbars defined in the active window.

DESCRIPTION

Return a Tcl list of the names of all the toolbars that have been created with gui_create_toolbar command for the specified window.

SEE ALSO

gui_create_toolbar(2)
gui_create_toolbar_item(2)
gui_delete_toolbar(2)
gui_delete_toolbar_item(2)
gui_hide_toolbar(2)
gui_show_toolbar(2)

gui_get_utable

Get various state about User Table(s) [default: return list of all table names].

SYNTAX

string **gui_get_utable**
-name *Table_Name*
-has_name
-tag
-window *Window_name*
-values *Column_Name*
-columns

Table_Name String
Window_Name String
Column_Name String

ARGUMENTS

-name *Table_Name*

A table name argument used with other argument switches below.

-has_name

Check if given table name given by the '-name' argument switch exist and return '1', else return '0'.

-tag

Get the table MetaData tag string for given table argument.

-window *Window_Name*

A UserTable window name argument used with other arguments below. If used alone it will return the current table name of the given window name.

-values *Column_Name*

Get the selected table cell value(s) in the given column for the given window argument. **Note**: Any column value(s) in the row of the selected cell(s) can be retrieved.

-columns

Get the selected table column name(s) for the given window argument.

DESCRIPTION

This command allows you to get all current loaded UserTable names.
Other options allow you to get various other table state.

EXAMPLES

The following example check to see if the UserTable 'my_table' is currently loaded.

```
shell> if { [gui_get_utable -name my_table -has_name] } {
         echo "yes my_table is loaded"
       }
```

The following example looks for the 'full_name' column in the current active UserTable and prints all the selected values in that column.

```
shell> # Get the current active UserTable window:
     set win [gui_get_current_window -mru -type UserTable]
     # Get the table name in that window:
     set table [gui_get_utable -window $win]

     if { $table != {} } {
      # Get the selected column names in that table:
      set columns [gui_get_utable -window $win -columns]
      # If the column name 'full_name' is selected then list the values:
      if { [lsearch -exact $columns full_name] != -1} {
        set values [gui_get_utable -window $win -values full_name]
        foreach v $values {
          echo "value: $v"
        }
      }
     }
```

SEE ALSO

gui_append_utable(2)
gui_close_utable(2)
gui_change_selection_utable(2)
gui_create_utable(2)
gui_export_utable(2)

gui_import_utable(2)
gui_open_utable(2)
gui_show_utable(2)
gui_write_utable(2)

gui_get_var

Get the value of a typed tcl variable that is created with gui_create_var.

SYNTAX

string **gui_get_var** -name *VarName*

VarName *String*

ARGUMENTS

-name *VarName*

VarName specifies the name of the typed tcl variable to retrieve the value of.

DESCRIPTION

This command retrieves the value of a typed tcl variable.

SEE ALSO

gui_create_var(2)
gui_set_var(2)
gui_remove_var(2)
gui_exist_var(2)

gui_get_vm

Retrieves attributes for a specified visual mode.

SYNTAX

string **gui_get_vm**
-name *identifier*
[-buckets]
[-title]
[-help_topic]
[-infotip]
[-discrete]
[-icon_file]
[-update_cmd]
[-netfilter]
[-float]
[-top_exaggeration]
[-mid_exaggeration]
[-bot_exaggeration]
[-show_only_pins_of_nets]
[-enable_bucket_delete]

Data Types

identifier string

ARGUMENTS

-name *identifier*

Specifies the name of the visual mode to be queried.

-buckets

Returns a list of bucket names. The bucket names are retrieved in rendering order, starting from the first to the last bucket rendered.

-title

Returns the title string.

-help_topic

Returns the help topic string.

-infotip

Returns the InfoTip string.

-discrete

Returns a true or false value that indicates whether the buckets of this visual mode are discrete or continuous. Only discrete buckets can be reordered. This attribute is set when this visual mode was created, and cannot be changed.

-icon_file

Returns the gui menu icon file.

-update_cmd

Returns the Tcl command that is executed when the visual mode is accessed. A typical use for this option is to update the state of the visual mode if its contents are likely to change under certain circumstances.

-netfilter

Returns net connection filtering.

-float

Returns a true or false value indicating whether the visual mode buckets have a floating point range.

-top_exaggeration

Returns the top bucket exaggeration.

-mid_exaggeration

Returns the middle bucket exaggeration.

-bot_exaggeration

Returns the bottom bucket exaggeration.

-show_only_pins_of_nets

Returns a true or false value that indicates whether the layout shows only pins of net objects in buckets.

-enable_bucket_delete

Returns a true or false value indicating whether the visual mode buckets can be deleted from context menu.

DESCRIPTION

The **gui_set_vm** command queries the option values of an existing visual mode.

EXAMPLES

To retrieve names of the buckets in the visual mode named "mycoloring1", use the following command:

prompt>**gui_get_vm -name mycoloring1 -buckets**

SEE ALSO

gui_create_vm(2)
gui_create_vmbucket(2)
gui_get_vmbucket(2)
gui_remove_vm(2)
gui_remove_vmbucket(2)
gui_set_vm(2)
gui_set_vmbucket(2)
gui_update_vm(2)

gui_get_vmbucket

Get attributes for Visual Mode Bucket

SYNTAX

string **gui_get_vmbucket** -vmname *mode_identifier*
-name *bucket_identifier*
[-infotip]
[-netfilter]
[-color]
[-pattern]
[-exaggeration]
[-number]
[-maxval]
[-minval]
[-visible]
[-title]
[-collection]
[-objcount]

string *mode_identifier*
string *bucket_identifier*

ARGUMENTS

-vmname *mode_idnetifier*

Specifies the visual mode to which the bucket is a member. The visual mode must already exist. To see the current list of defined visual modes use the gui_list_vm command.

-name *bucket_identifier*

Specifies the name of the visual mode bucket to be queried. To see the list of buckets associated with a specific visual mode use the gui_get_vm command with the -buckets option.

-infotip

Return infotip string.

-netfilter *net_filter*

Return net connection filtering.

-color

Return bucket rendering color.

-pattern

Return bucket rendering fill pattern.

-exaggeration

Return bucket min pixel exaggeration value.

-number

Return bucket display number.

-maxval

Return bucket maximum value.

-minval

Return bucket minimum value.

-visible

Return visibility.

-title

Return bucket title.

-collection

Return object collection.

-objcount

Return the number of objects in the bucket.

DESCRIPTION

The gui_get_vmbucket command queries an instance of a visual mode bucket for the specified option values. The values are returned in a list of option-value pairs.

EXAMPLES

Query the current color for the bucket named "cat1" in the visual mode "foo":

shell> **gui_get_vmbucket -vmname foo -name cat1 -color**

Query the current color and pattern for the bucket named "cat1" in the visual mode "foo":

shell> **gui_get_vmbucket -vmname foo -name cat1 -color -pattern**

SEE ALSO

gui_create_vm(2)
gui_create_vmbucket(2)
gui_get_vm(2)
gui_set_vmbucket(2)
gui_list_vm(2)
gui_remove_vm(2)
gui_remove_vmbucket(2)
gui_set_vm(2)
gui_update_vm(2)

gui_get_window_ids

Get a list of window ids

SYNTAX

ids **gui_get_window_ids**
[-parent *window_id*]
[-type *window_type*]

Data Types

window_id string
window_type string

ARGUMENTS

-parent *window_id*

Specifies the toplevel window id to get child view window ids from.

-type *window_type*

Specifies that the windows id returned should be restricted to those of the specified type.

Note: as a type can be only be associated with either a toplevel window or a child view window (not both) the type implies whether toplevel or child view window ids are returned.

If this option is not specified then either all toplevel window ids are returned (*-parent* option not specified) or all child view window ids of a specified toplevel window id are returned (*-parent* option specified).

RETURNS

ids

Returned list of window ids.

DESCRIPTION

This command gets a list of the ids of all toplevel windows, all toplevel windows of a specified type, all child view windows of a specified type, or all child view windows of a specified type in a specified toplevel window.

EXAMPLES

The following example will create a toplevel window and a child layout window. Then this command will be used to retrieve the ids of the existing windows.

```
shell> set top [gui_create_window -type TopLevel]
shell> set cons [gui_create_window -type Layout -parent $top]
shell> set ids [gui_get_window_ids]   # return list containing values of $top and value of $cons.
shell> set childs [gui_get_window_ids -parent $top]   # return value of $cons.
shell> set layout_instances [gui_get_window_ids -type Layout] # return value of $cons.
```

SEE ALSO

gui_close_window(2)
gui_exist_window(2)
gui_show_window(2)
gui_get_window_types(2)
gui_create_window(2)
gui_set_active_window(2)
gui_get_current_window(2)

gui_get_window_pref_categories

Get list of preference categories for object specified by window

SYNTAX

categories **gui_get_window_pref_categories**
{ -window *window_id* | -window_type *window_type* }

Data Types

window_id string
window_type string

ARGUMENTS

-window *window_id*

Specifies the name of the window that the preference categories will be retrieved from. This option is mutually exclusive with the -window_type option.

-window_type *window_type*

Specifies the name of the window type that instances of the type will have the preference categories be retrieved from. This option is mutually exclusive with the -window option.

RETURNS

categories

The returned preference categories.

DESCRIPTION

This command retrieves the list of preference categories for a window or window type.

EXAMPLES

The following example gets the preference categories for the TopLevel.1 window.

```
shell> ser cats [gui_get_window_pref_categories -window TopLevel.1]
```

SEE ALSO

gui_set_window_pref_key(2)
gui_get_window_pref_value(2)
gui_exist_window_pref_key(2)
gui_remove_window_pref_key(2)
gui_get_window_pref_keys(2)
gui_get_window_pref_attribute(2)
gui_set_window_pref_attributes(2)
gui_create_pref_category(2)
gui_set_pref_value(2)
gui_get_pref_value(2)
gui_get_pref_value_type(2)
gui_remove_pref_key(2)
gui_get_pref_categories(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_get_window_pref_keys

Get list of preference categories for object specified by window

SYNTAX

categories **gui_get_window_pref_keys**
{ -window *window_id* | -window_type *window_type* }
[-category *category*]

Data Types

window_id string
window_type string
category string

ARGUMENTS

-window *window_id*

Specifies the name of the window that the preference categories will be retrieved from. This option is mutually exclusive with the -window_type option.

-window_type *window_type*

Specifies the name of the window type that instances of the type will have the preference categories be retrieved from. This option is mutually exclusive with the -window option.

-category *category*

Specifies the category that the key belongs to. If not specified, a default category will be assigned.

RETURNS

categories

The returned preference categories.

DESCRIPTION

This command retrieves the list of preference categories for a window or window type.

EXAMPLES

The following example gets the preference categories for the TopLevel.1 window.

```
shell> ser cats [gui_get_window_pref_keys -window TopLevel.1]
```

SEE ALSO

gui_set_window_pref_key(2)
gui_get_window_pref_value(2)
gui_exist_window_pref_key(2)
gui_remove_window_pref_key(2)
gui_get_window_pref_categories(2)
gui_get_window_pref_attribute(2)
gui_set_window_pref_attributes(2)
gui_create_pref_category(2)
gui_set_pref_value(2)
gui_get_pref_value(2)
gui_get_pref_value_type(2)
gui_remove_pref_key(2)
gui_get_pref_categories(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_get_window_pref_value

Get preference value for object specified by window or window type

SYNTAX

value **gui_get_window_pref_value**
{ -window *window_id* | -window_type *window_type* }
[-category *category*]
-key *key*

Data Types

window_id string
window_type string
category string
key string

ARGUMENTS

-window *window_id*

Specifies the name of the window that the preference will be retrieved from. This option is mutually exclusive with the -window_type option.

-window_type *window_type*

Specifies the name of the window type that instances of the type will have the preference be retrieved from. This option is mutually exclusive with the -window option.

-category *category*

Specifies the category to search for the key. If not specified, a default category will be used.

-key *key*

Specifies the key to retrieve the value from.

RETURNS

value

The value of the key.

DESCRIPTION

This command retrieves the value of a preference key from the specified key name and category of the specified window or window type. If a preference is set on a window type, then all instances of that type will inherits the preference, so the value can be set on a type and retrieved from any instance of that type (or derived from that type).

EXAMPLES

The following example creates a preference for a window and then uses the **gui_get_window_pref_value** command to retrieve the value for the preference.

```
shell> gui_set_window_pref_key -window Console.1 -key test -value_type integer -value 201
shell> set x [gui_get_window_pref_value -window Console.1 -key test]
shell> if {$x == 201} { echo "success" }
```

SEE ALSO

gui_set_window_pref_key(2)
gui_exist_window_pref_key(2)
gui_remove_window_pref_key(2)
gui_get_window_pref_categories(2)
gui_get_window_pref_keys(2)
gui_get_window_pref_attribute(2)
gui_set_window_pref_attributes(2)
gui_create_pref_category(2)
gui_set_pref_value(2)
gui_get_pref_value(2)
gui_get_pref_value_type(2)
gui_remove_pref_key(2)
gui_get_pref_categories(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_get_window_presets

Gets list of preset name for object specified by window_type

SYNTAX

string **gui_get_window_presets**
-window_type *window_type*
[-system]
[-shared]

Data Types

window_type string

ARGUMENTS

-window_type *window_type*

Specifies the name of the window type that the presets be retrieved from.

-system

Return system presets. This option is mutually exclusive with the -shared option.

-shared

Return shared presets. This option is mutually exclusive with the -system option.

DESCRIPTION

This command retrieves the list of presets for a window type. This command has been deprecated, please use gui_get_presets instead.

EXAMPLES

The following example gets the system presets for the Layout.

prompt> **gui_get_window_presets -window_type Layout -system**

SEE ALSO

gui_get_presets(2)
gui_set_preset(2)
gui_set_window_preset(2)

gui_get_window_types

Get a list of window types

SYNTAX

types **gui_get_window_types**
[-type *token*]

Data Types

token string

ARGUMENTS

-type *window_types*

Specifies whether to return toplevel windows ('toplevel') or child view windows ('child').

If toplevel window types are specified then the first type is guaranteed to be the default toplevel window type.

RETURNS

types

Returned list of window types.

DESCRIPTION

This command returns a list of existing window types. Instances of these types can be instantiated with the **gui_create_window** command. The returned list can be restricted to just toplevel windows or child view windows by specifying the *-type* option.

EXAMPLES

The following examples illustrate some usages of the command.

```
shell> gui_get_window_types
shell> gui_get_window_types -type toplevel
shell> gui_get_window_types -type child
```

SEE ALSO

gui_close_window(2)
gui_exist_window(2)
gui_show_window(2)
gui_create_window(2)
gui_get_window_ids(2)
gui_set_active_window(2)
gui_get_current_window(2)

gui_group_charts_plots

Group plots in charts view

SYNTAX

status **gui_group_charts_plots**
-view *view_name*
[-x1x2]
[-y1y2]
[-overlay]
[*plots*]

Data Types

view_name string
plot_id_list list

ARGUMENTS

-view *view_name*

Charts View to group.

-x1x2

This option specifies that two plots are connected such that they have a shared y axis range but two independent x axes ranges.

If the *-overlay* option is used then the two plots are overlaid and the X axis of the first plot is placed on the bottom and the X axis of the second plot is placed on the top. Note: To ensure the plots are overlaid correctly the two plots are forced to the same area of the view (usually the whole view).

If the *-overlay* option is not used then the two plots are not overlaid and can be placed anywhere in the view. For best results horizontal stacking is recommended so the shared y axes are placed next to each other.

There must be exactly two plots specified to the *-plots* option.

This option cannot be combined with the *-y1y2* option.

-y1y2

This option specifies that two plots are connected such that they have a shared x axis range but two independent y axes ranges.

If the *-overlay* option is used then the two plots are overlaid and the Y axis of the first plot is placed on the bottom and the Y axis of the second plot is placed on the top. Note: To ensure the plots are overlaid correctly the two plots are forced to the same area of the view (usually the whole view).

If the *-overlay* option is not used then the two plots are not overlaid and can be placed anywhere in the view. For best results vertical stacking is recommended so the shared x axes are placed next to each other.

There must be exactly two plots specified to the *-plots* option.

This option cannot be combined with the *-x1x2* option.

-overlay

This option specifies that the plots have shared x and/or y ranges and are drawn on top of each other so they can share a single key and title.

If the *-x1x2* or *-y1y2* option are specified then only the x or y axis range is shared (See descriptions of these options above).

If neither of the *-x1x2* or *-y1y2* options are specified then both the x and y axis ranges are shared. In this case two or more plots can be specified.

plots

List of plots to group.

For *x1x2* or *y1y2* options this must be two plots. For *-overlay* option, without the *x1x2* and *y1y2* options, this must be two or more plots.

DESCRIPTION

Group related plots in a view.

Grouped plots can share x or y axis ranges and can be overlaid.

Sharing x and/or y axis is useful when you want to compare data across two plots. Overlay is useful when you want to display multiple plot types with the same data e.g. a combined barchart and xy plot (line plot).

EXAMPLES

The following example groups two plots so they are overlaid and share a common x axes.

```
shell> gui_group_charts_plots -view $view -y1y2 -overlay -plots [list $plot1 $plot2]
```

SEE ALSO

gui_create_charts_plot(2)
gui_place_charts(2)

gui_hide_palette

Hides the specified palette.

SYNTAX

status **gui_hide_palette**
{ -name *palette_id* | -type *palette_type* | -all }
[-parent *window_id*]

Data Types

palette_id string
palette_type string
window_id string

ARGUMENTS

-name *palette_id*

Specifies the palette name id or title.

If the *-parent* option is specified, only palettes in the specified parent toplevel window are hidden. If the *-parent* option is not specified, all palettes of this name in all toplevel windows are hidden.

This option precludes the *-type* and *-all* options.

-type *palette_type*

Specifies the palette type id. Any matching palette of this type will be hidden.

If the *-parent* option is specified, only palettes in the specified parent toplevel window are hidden. If the *-parent* option is not specified, all palettes of this type in all toplevel windows are hidden.

This option precludes the *-name* and *-all* options.

-all

Hides all palettes.

If the *-parent* option is specified, only palettes in the specified parent toplevel window are hidden. If the *-parent* option is not specified, all palettes in all toplevel windows are hidden.

This option precludes the *-name* and *-type* options.

-parent *window_id*

Specifies the parent toplevel window to hide the palette type in.

Note: this option is only applicable if the *-type* option is specified.

DESCRIPTION

This command hides the specified palettes.

EXAMPLES

The following example hides all Layer palettes:

shell> gui_hide_palette -type Layout

SEE ALSO

gui_show_palette(2)

gui_hide_toolbar

Hides the specified toolbar or all the toolbars in the specified window or for a window type.

SYNTAX

void **gui_hide_toolbar**
-toolbar *tool_bar_name* | -all
[-window *window_id*]
[-window_type *window_type*]

Data Types

tool_bar_name string
window_id string
window_type string list

ARGUMENTS

-toolbar *tool_bar_name*

Specifies the toolbar you want to hide. The **-toolbar** option and the **-all** option are mutually exclusive.

-all

Hides all the toolbars in the specified window. The **-all** option and the **-toolbar** option are mutually exclusive.

-window *window_id*

Specifies the window in which you want to hide the specified toolbar or all toolbars. This option is mutually exclusive with the **-window_type** option. If both **-window** and **-window_type** options are omitted, then by default the tool hides the toolbar or toolbars in the active window.

-window_type *window_type*

Specifies the window types in which you want to hide the specified toolbar or all toolbars. All existing windows of the given types will be affected. Toolbar visibility may be set on a window type before a window of the type exists. This option is mutually exclusive with the **-window** option. If both **-window** and **-window_type** options are omitted, then by default the tool hides the toolbar or toolbars in the active window.

DESCRIPTION

This command hides either all the toolbars or the toolbar with the specified name in the window with the specified window ID, or in

all windows of the given window type.

Toolbar visibility set by this command will override any visibility setting that has been saved and restored from previous sessions. The last toolbar visibility set by this command or by **gui_show_toolbar** will be saved and restored on next invocation of the tool if save and restore has not been disabled.

EXAMPLES

The following example hides the File toolbar in the active window:

```
prompt> gui_hide_toolbar -toolbar File
```

The following example hides all the toolbars in the layout window named LayoutWindow.1:

```
prompt> gui_hide_toolbar -all -window LayoutWindow.1
```

The following example hides the File toolbar in all TimingWindow type windows:

```
prompt> gui_hide_toolbar -all -window_type TimingWindow
```

The following example hides all the toolbars in all window types:

```
prompt> gui_hide_toolbar -all -window_type [gui_get_window_types -type toplevel]
```

SEE ALSO

gui_create_toolbar(2)
gui_delete_toolbar(2)
gui_create_toolbar_item(2)
gui_delete_toolbar_item(2)
gui_show_toolbar(2)
gui_get_toolbar_names(2)
gui_get_window_types(2)

gui_highlight_nets_of_selected

Highlight the nets of the selected object in the current highlight color.

SYNTAX

status **gui_highlight_nets_of_selected**
[-flylines]

ARGUMENTS

-flylines

Specifies that the net should always be displayed on the layout using flylines, even if it is routed.

DESCRIPTION

The **gui_highlight_nets_of_selected** command highlights the nets of the objects currently selected in the GUI. Typically this is used in conjunction with the Layout view to view the highlighted nets, and is accessed via the Highlight menu item on the Layout Window. This command adds to any existing highlights and the highlighting will remain until cleared by the user or the objects referenced in the highlight are removed. If no objects are selected this command will not do anything.

The layout view renders the net using its routing if the net has routing, and if it does not the net is rendered using a flyline. However if the -flyline option is specified the layout will always render the highlighted net using the flylines between its pins.

EXAMPLES

Highlight the nets of the selected objects.

prompt> **gui_highlight_nets_of_selected**

SEE ALSO

gui_change_highlight(2)
change_selection(2)

gui_import_utable

Create a UserTable and import records from a value file or a report.

SYNTAX

string **gui_import_utable**
-file *File_Name* [-tsv_mode] [-ssv_mode]
[-name *Table_Name*]
[-tsv_mode]
[-ssv_mode]
[-fold]
[-report_type *Report_Type_Name*]
[-meta_obj *MetaObj_Name*]
[-replace]

Table_Name String
File_Name String
Report_Type_Name String
MetaObj_Name String

ARGUMENTS

-file *File_Name*

The name of the CSV value file from which to import and fill the user table with.

-name *Table_Name*

The name for the user table created. By default, the name takes the name of the file minus the extension.

-tsv_mode

This flag changes the default delimiter from the comma char to the tab char.

-ssv_mode

This flag changes the default delimiter from the comma char to the semicolon char.

-fold

This flag causes extra row data past the number of columns specified to fold into the last column. The default is to discard extra data.

-report_type

This flag causes the reader to interpret the file as a report file and convert it into a user table. *Note:* Currently only the 'report_constraint -all_violators' report can be converted.

-meta_obj *MetaObj_Name*

Optional MetaData object name used to define table meta data. See the command gui_set_utable_meta for more information.

-replace

Optional flag that will replace the given table name, if it already exists, with the imported table data.

DESCRIPTION

This command creates a new UserTable in memory and imports the data from a CSV/TSV/SSV value file. The name of the table will take the base name of the file unless a name is provided.

The first line of the file is assumed to be a header line that defines the column names. In addition, column names can have postfixes that describe the data format of that column.

Supported column name postfixes are:

:string
:int
:double
:cell
:net
:port
:pin
:point
:endpoint
:file

EXAMPLES

The following example creates a new UserTable called 'data' and imports the values from the CSV file 'data.csv'.

shell> **gui_import_utable -file data.csv**

The following example creates a new UserTable, names it 'my_table' called 'data' and imports the values from the CSV file 'data.csv'. It also instructs the import to fold any extra data found in the row into the last column defined.

shell> **gui_import_utable -file data.csv -name my_table -fold**

SEE ALSO

gui_append_utable(2)
gui_change_selection_utable(2)
gui_close_utable(2)
gui_create_utable(2)
gui_export_utable(2)
gui_get_utable(2)
gui_set_utable_meta(2)
gui_open_utable(2)
gui_show_utable(2)
gui_write_utable(2)

gui_list_attrgroups

Lists attribute group information for a specified object type or all object types.

SYNTAX

status **gui_list_attrgroups**
-class *design_object*
-name *name*
[-all]
[-tcl]
[-full]
[-attr_list]

Data Types

design_object string
name string

ARGUMENTS

-class *design_object*

Specifies the type of design object.

-name *name*

Specifies the name of the attribute group to be listed for the specified object type or for all object types.

-all

Lists the attribute groups for all object types. This option and the -class option are mutually exclusive.

-tcl

Lists the attribute group information in a tool command language (Tcl) format that is suitable for Tcl processing.

-full

Specifies the full Tcl output format, which reports all possible group information.

-attr_list

Lists attributes only.

DESCRIPTION

The **gui_list_attrgroups** command lists attribute group information for the specified object type or for all object types. Use the **-tcl** option to get results that are suitable for Tcl processing.

EXAMPLES

prompt> **gui_list_attrgroups -class Cell -tcl**
"Hierarchy" "Edit"

prompt> **gui_list_attrgroups -class Cell -name "Hierarchy" -tcl -attr_list**
{Name} {Hierarchical} {Owning Cell} {isPhysConstraint} {Hard Macro} ... {FullName}

prompt> **gui_list_attrgroups -all -tcl**
Cell { "Hierarchy" "Edit" } Net { "Timing" } Pin { "Timing" }

prompt> **gui_list_attrgroups -class Cell -tcl -attr_list**
{ "Hierarchy" { {Name} {Hierarchical} {Owning Cell} ...{FullName} } }
{ "Edit" { {Name} {isPhysConstraint} {Orientation} ... {FullName} } }

prompt> **gui_list_attrgroups -class Cell**
Cell, "Hierarchy", { {Name} {Hierarchical} {Owning Cell} {isPhysConstraint} ... } ;
Cell, "Edit", { {Name} {isPhysConstraint} {Orientation} {Location} ... } ;
The fields displayed above are in the following order:
Class, Group Name, Attributes list;

SEE ALSO

gui_delete_attrgroup(2)
gui_list_attrgroups(2)
gui_update_attrgroup(2)

gui_list_category_rules

Lists all category rules except built-in rules by default.

SYNTAX

list **gui_list_category_rules**
[-all | -names *rule_names_list*]
[-format script | tcl_list]

Data Types

rule_names_list list

ARGUMENTS

-all

Lists all category rules, including built-in rules.

This option and the **-names** option are mutually exclusive. If you do not specify either of these options, the command lists all category rules except the built-in rules.

-names *rule_names_list*

Specifies the names of the category rules to be listed.

This option and the **-all** option are mutually exclusive. If you do not specify either of these options, the command lists all category rules except the built-in rules.

-format script | tcl_list

Specifies the output format in which the category rules are listed. The default output format is **script**.

When the **script** format is used, the category rules are listed as a Tool Command Language (Tcl) script of **gui_create_category_rule** commands that you can replay. In this case, the rules are streamed to the output stream. You can redirect the output by using the **redirect** command, for example, if you want to capture the script output in a file to use again later. When the **script** format is used, the command result is an empty string.

If the **tcl_list** format is specified, the category rules are listed in an "array set compatible" Tcl list format. In this case, the command result is a Tcl list.

DESCRIPTION

This command lists category rules that have already been created by the **gui_create_category_rule** command during the current run of the tool.

When you run this command without specifying either the **-all** option or the **-names** option, the command lists all category rules except built-in rules. Specify the **-all** option to list all category rules, including the built-in rules. Specify the **-names** option to list individual rules by name.

EXAMPLES

The following example lists all category rules, including the built-in rules:

```
prompt> gui_list_category_rules -all
gui_create_category_rule  -name Pathgroups  -builtin \
  -category <path_group.full_name>
gui_create_category_rule  -name Session  -builtin \
  -category <session_name>
gui_create_category_rule  -name {Path Type}  -builtin \
  -category <path_type>
  ...
  ...
```

The following example lists all category rules except the built-in rules:

```
prompt> gui_create_category_rule  -name rule1 -category <session_name>
rule1
prompt> gui_create_category_rule  -name rule2 -category <path_group.full_name>
rule2
prompt> gui_list_category_rules
gui_create_category_rule  -name rule1 \
  -category <session_name>
gui_create_category_rule  -name rule2 \
  -category <path_group.full_name>
```

The following example redirects the script output to a file that you can source during a subsequent run of the tool:

```
prompt> redirect -file /remote/dir1/rules.tcl {gui_list_category_rules}
prompt>
```

The following example uses the **-names** option to list only the rules named rule1 and rule3:

```
prompt> gui_create_category_rule  -name rule1 -category <session_name>
rule1
prompt> gui_create_category_rule  -name rule2 -category <path_group.full_name>
rule2
prompt> gui_create_category_rule  -name rule3 -category <startpoint.full_name>
rule3
prompt> gui_list_category_rules -names {rule1 rule3}
gui_create_category_rule  -name rule1 \
  -category <session_name>
gui_create_category_rule  -name rule3 \
  -category <startpoint.full_name>
```

The following example uses the **tcl_list** output format to query the category specification of the rule named rule1:

```
prompt> gui_create_category_rule  -name rule1 -category <session_name>
```

```
rule1
prompt> gui_create_category_rule  -name rule2 -category <path_group.full_name>
rule2
prompt> array set rules [gui_list_category_rules -format tcl_list]
Information: Defining new variable 'rules'. (CMD-041)
prompt> array set rule1_options $rules(rule1)
Information: Defining new variable 'rule1_options'. (CMD-041)
prompt> set rule1_cat_spec $rule1_options(category)
Information: Defining new variable 'rule1_cat_spec'. (CMD-041)
<session_name>
```

SEE ALSO

gui_create_category_rule(2)
gui_remove_category_rules(2)

gui_list_cell_block_marks

Lists the cell names and block mark values for cells that are marked in the design.

SYNTAX

list **gui_list_cell_block_marks**

ARGUMENTS

This command has no arguments

DESCRIPTION

This command lists the cell names and block mark values for all cells that are marked in the design. The listing is sorted by the full names of the marked cells.

EXAMPLES

The following example sets block marks on two hierarchical cell instances, and then lists the names of all the marked cells and their block marks:

```
prompt> gui_remove_cell_block_marks -all
prompt> gui_set_cell_block_marks I_TOP/I_ALU  ALU
prompt> gui_set_cell_block_marks I_TOP/I_REG_FILE  REG_FILE
prompt> gui_list_cell_block_marks
I_TOP/I_ALU    ALU
I_TOP/I_REG_FILE    REG_FILE
```

SEE ALSO

gui_set_cell_block_marks(2)
gui_get_cell_block_marks(2)

gui_remove_cell_block_marks(2)

gui_list_vm

List Current Visual Modes

SYNTAX

string **gui_list_vm**

ARGUMENTS

DESCRIPTION

The gui_list_vm command will return a list of existing visual modes.

EXAMPLES

To get the names of all available visual modes, use the following command:

shell> **gui_list_vm**

SEE ALSO

gui_create_vm(2)
gui_create_vmbucket(2)
gui_get_vm(2)
gui_set_vm(2)
gui_get_vmbucket(2)
gui_remove_vm(2)
gui_remove_vmbucket(2)
gui_set_vmbucket(2)
gui_update_vm(2)

gui_load_area_net_connection_vm

Loads data for the Net Connection in Area visual mode.

SYNTAX

status **gui_load_area_net_connection_vm**
-area {*llx lly urx ury*}
[-type {power ground clock signal tie_high tie_low}]
[-hier]

Data Types

llx float
lly float
urx float
ury float

ARGUMENTS

-area {*llx lly urx ury*}

Specifies the area to analyze for net connectivity.

-type {power ground clock signal tie_high tie_low}

Specifies one or more types of nets to show. Possible types are power, ground, clock, signal, tie_high, and tie_low. This option takes types as a list of strings.

-hier

Specifies that the analysis should include subhierarchies. If this option is not specified, the tool analyzes only nets in the current top-level design.

DESCRIPTION

The command generates visual mode of net connections in the specified area. Net connections are categorized as follows:

- local connection: Nets have all their pins within the specified area.
- global connection: nets have at least one pin outside of the specified area, and one pin inside of specified area.
- path-through connection: nets have all their pins outside of specified area.

Nets in the visual mode are displayed based on layout view setting of net.

EXAMPLES

The following example shows clock and signal types of net connections in the specified area.

```
prompt> gui_load_area_net_connection_vm -area \
  {550.710 1840.430 1247.100 2150.640} -type {clock signal}
```

SEE ALSO

None

gui_load_cell_density_mm

Loads the data for cell density map mode.

SYNTAX

status **gui_load_cell_density_mm**
[-area *area*]

Data Types

area list

ARGUMENTS

-area *area*

Analyzes area for cell density. If area is not given, the default area is whole design area.

DESCRIPTION

The command generates the map mode view of cell density in a given area. Each grid is colored based on the percentage of occupied area of cells in the grid.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows cell density in the specified area.

```
prompt> gui_load_cell_density_mm -area \
{284.820 390.670 1202.790 954.120}
```

SEE ALSO

None

gui_load_cell_slack_vm

Loads the data for cell slack visual mode.

SYNTAX

status **gui_load_cell_slack_vm**
[-scenario *scenario*]
[-analysis *approach*]
[-num_bins *count*]
[-bin_range *range*]
[-lower_bound *value*]
[-lower_bound_strict]
[-upper_bound *value*]
[-upper_bound_strict]
[-cells *cell_list*]
[-color_scheme *scheme*]

Data Types

scenario string
approach string
count integer
range float
value float
cell_list list
scheme string

ARGUMENTS

-scenario *scenario*

Specifies the scenario for the visual mode. By default, current scenario is used.

-analysis setup | hold

Specifies the analysis approach for the visual mode. The alternatives are setup and hold. By default, the analysis approach is setup.

-num_bins *count*

Specifies the number of bins to create.

-bin_range *range*

Specifies the range of each bin to create when the number of bins is not specified.

-lower_bound *value*

Specifies the lower bound value for the bins. Values below this bound are placed in a separate bin.

-lower_bound_strict

Uses the lower bound as an exact minimum limit on the bound. Values below the lower bound are ignored.

-upper_bound *value*

Specifies the upper bound value for the bins. Values above this bound are placed in a separate bin.

-upper_bound_strict

Uses the upper bound as an exact maximum limit on the bound. Values above the upper bound are ignored.

-cells *cell_list*

Specifies the cells to be used in the visual mode. If this option is not specified, all the cells in the design are used by default.

-color_scheme temperature | gradient

Specifies the bucket coloring approach in the visual mode. The alternatives are temperature scale and gradient. By default, the color scheme is temperature scale.

DESCRIPTION

The command generates the visual mode view of cell slack. Each cell is colored based on the color of the bin it falls into.

EXAMPLES

The following example will place all the cells in the design into 8 bins.

```
prompt> gui_load_cell_slack_vm -num_bins 8 \
-lower_bound 0
```

The following example will place all the currently selected cells into 8 bins.

```
prompt> gui_load_cell_slack_vm -num_bins 8 \
-lower_bound 0 -cells [get_selection] -scenario [current_scenario]
```

SEE ALSO

None

gui_load_clock_trunk_planning

Starts a GUI clock trunk planning session.

SYNTAX

gui_load_clock_trunk_planning
[-clocks *clocks*]
[-include_generated]
[-self]
[-show_timing_flylines]

Data Types

clocks list

ARGUMENTS

-clocks *clocks*

Specifies the clocks to plan. By default, planning is performed for all clocks and all virtual clocks and generated clocks are skipped.

-include_generated

Includes generated clocks when get auto clocks. This option should not be used with the **-clocks** option.

-self

Ignore master clock portion for generated clock.

-show_timing_flylines

Generates data for timing flylines. This option requires a timing update.

DESCRIPTION

This command starts a clock trunk planning session for a given clock. To view the clock trunk in the layout window, open the GUI clock trunk planning flylines tool. The flylines tool identifies the clock trunk and displays the clock trunk annotations in the layout.

The clock trunk is identified as follows:

- The clock source pins are part of the clock trunk.

- If a certain pin is part of the clock trunk, then all pins on all paths from the clock source pin to that certain pin are implicitly part of the clock trunk
- All user-defined clock trunk endpoints are part of the clock trunk.
- All first-level block conns/ports are part of the clock trunk, unless there is already a user-defined clock trunk endpoint upstream (for example, somewhere on all paths from the clock root to here).
- For all pins in the clock trunk which have less than *fanout_limit* number of fanouts, the tool recursively adds fanout. However this is not done for clock-trunk leafs that are a user-defined clock trunk endpoint or a first-level block conn/port. (Note that nonleafs which fall into these categories are still expanded).
- The leaf pins of the resulting clock trunk are called clock trunk end points.

EXAMPLES

Start a planning session on all clocks which enter some block.

```
prompt> gui_load_clock_trunk_planning
```

Start a planning session for clock CLK.

```
prompt> gui_load_clock_trunk_planning -clock CLK
```

SEE ALSO

create_clock_buffer(2)
set_clock_trunk_endpoints(2)
synthesize_clock_trees(2)

gui_load_hierarchy_vm

Loads the data for hierarchy visual mode.

SYNTAX

status **gui_load_hierarchy_vm**
[-clear]
[-level *level*]
[-objects *object_list*]
[-hierarchical]

Data Types

level integer
object_list list

ARGUMENTS

-clear

Removes the coloring from all hierarchy levels.

-level *level*

Colors all the cells at a particular hierarchy level.

-objects *object_list*

Colors specific hierarchy objects. If objects are not provided, the tool colors cells in the entire design. if *plan.place.hierarchy_by_name* is *true*, the option accepts flat hier name bound objects.

-hierarchical

Specifies that subhierarchies are included in the analysis. If this option is not specified, the tool analyzes only cells in the current top-level design.

DESCRIPTION

This command loads the data for hierarchy visual mode. The hierarchy visual mode provides a high-level view of the placement quality of logic blocks and hierarchical cells in your physical design. This visual mode can be used to color all the cells on a particular hierarchy level or just the specified hierarchical cells.

EXAMPLES

The following example clears the hierarchy visual mode.

```
prompt> gui_load_hierarchy_vm -clear
```

The following example colors all the cells at hierarchical level 2.

```
prompt> gui_load_hierarchy_vm -level 2
```

The following example colors all the objects in the selection.

```
prompt> gui_load_hierarchy_vm -objects [get_selection]
```

SEE ALSO

None.

gui_load_imported_path_pins_vm

Loads the data for imported path pins visual mode.

SYNTAX

status **gui_load_imported_path_pins_vm**
[-timing_report *file_name*]
[-timing_text *report_text*]
[-draw_path *path_type*]
[-from *path_number*]
[-to *path_number*]
[-clear]

Data Types

file_name	string
report_text	string
path_type	string
path_number	integer

ARGUMENTS

-timing_report *file_name*

Generates the imported path pins visual mode from the timing report file. The file could be plain text or compressed(.gz).

-timing_text *report_text*

Generates imported path pins visual mode from the timing report text.

-draw_path *path_type*

Specifies the type to draw the path. The type value can be routes, flylines or both. The defult value is flylines. When specifies routes or both, the timing report should include input pins.

-from *path_number*

Generates imported path pins visual mode start from the specified path number.

-to *path_number*

Generates imported path pins visual mode ends with the specified path number.

-clear

Removes coloring from all pins and flylines.

DESCRIPTION

Loads the data for imported path pins visual mode. After importing timing path data from a timing report, the visual mode shows pins and the connectivity in one or more timing paths in Layout window. This visual mode colors the pins and flylines with a different color for each imported timing path.

EXAMPLES

The following example will load the timing report file "/remote/misc16/timing_report" to generate the imported path pins visual mode.

```
prompt> gui_load_imported_path_pins_vm \
  -timing_report "/remote/misc16/timing_report"
```

SEE ALSO

None

gui_load_path_analyzer_flylines

Loads the data for path analyzer flylines palette.

SYNTAX

status **gui_load_path_analyzer_flylines**

Data Types

None

ARGUMENTS

None

DESCRIPTION

The command generates the flylines data of path analyzer palette. Each timing path flyline is colored based on either its connection count in connection mode, or its timing effort in timing mode.

EXAMPLES

The following example reloads the timing path flylines data in the design.

prompt> **gui_load_path_analyzer_flylines**

SEE ALSO

None

gui_load_pin_density_mm

Loads the data for pin density map mode.

SYNTAX

status **gui_load_pin_density_mm**
[-area *area*]

Data Types

area list

ARGUMENTS

-area *area*

Area to be analyzed for pin density. By default, the command uses the entire design area.

DESCRIPTION

The command generates map mode view of pin density in a given area. The grids are colored based on the number of pins within the grid.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows pin density in the specified area.

```
prompt> gui_load_pin_density_mm -area \
  {284.820 390.670 1202.790 954.120}
```

SEE ALSO

None

gui_load_power_density_mm

Loads the data for power density map mode.

SYNTAX

status **gui_load_power_density_mm**
[-area *{{x1 y1} {x2 y2}}*]
-scenario *scenario_name*

Data Types

x1 float
y1 float
x2 float
y2 float
scenario_name string

ARGUMENTS

-area *{{x1 y1} {x2 y2}}*

Limits the report scope to the specified area. This argument is optional. By default, the entire design area is used.

-scenario

Specifies the scenario to be reported. This option is required.

DESCRIPTION

This command generates the data for the power density map mode. The design layout area, or user specified area, is split into grids. The grids dimension is set in the mapmode panel. The power contribution for cells that intersect the grid is proportional to the area within each grid sector.

Multicorner-Multimode Support

This command performs its analysis for specified scenario. The **-scenario** argument is required.

EXAMPLES

The following example reloads the power density map mode data from the "default" scenario.

prompt> **gui_load_power_density_mm -scenario default**

The following example reloads the power density map mode data within the specified area from the "sc1.BC" scenario.

prompt> **gui_load_power_density_mm -scenario sc1.BC -area { {10.5 0.50} {300.0 129.80} }**

SEE ALSO

None.

gui_load_routing_guide_vm

Loads the data for routing guides by type visual mode.

SYNTAX

status **gui_load_routing_guide_vm**
-types *type_list*
[-hierarchical]

Data Types

type_list list

ARGUMENTS

-types *type_list*

Specifies the types of routing guide to load in the visual mode. This option is required. The supported types are:

access_preference
default
design_boundary_blockage
forbidden_preferred_grid_extension
max_patterns
metal_cut_allowed
pin_access
river_routing
single_row_via_ladder_pattern_must_join_allowed
standard_cell_region
switched_direction_only

-hierarchical

Searches for routing guides level-by-level relative to the current instance. if this option is not specified, the tool loads only routing guide in the current top-level design.

DESCRIPTION

The command generates the visual mode view of routing guide. Each routing guide is colored based on its type.

EXAMPLES

The following example will color routing guides which type is access_preference or metal_cut_allowed.

prompt> **gui_load_routing_guide_vm -types {access_preference metal_cut_allowed}**

SEE ALSO

None

gui_load_rp_group_net_connectivity_vm

Loads the data for the relative placement (RP) net connection visual mode.

SYNTAX

status **gui_load_rp_group_net_connectivity_vm**
-clct *rp_group_collection*
[-type {clock signal}]

Data Types

rp_group_collection collection

ARGUMENTS

-clct *rp_group_collection*

Specifies the collection of RP groups to load.

-type {clock signal}

Specifies the types of nets to show. Valid signal types are *clock* and *signal*. This option takes the types as a list of strings.

DESCRIPTION

This command generates visual mode of net connections of a specified RP group collection. Net connections are categorized as follows:

- local connection: Nets that only connect to cells inside the RP group collection.
- global connection: Nets that connect to the cells outside the given RP group collection.

  Nets in the visual mode are displayed based on the layout view setting of net.

EXAMPLES

The following example shows signal net connections for the RP group collection name test1.

```
prompt> gui_load_rp_group_net_connectivity_vm -type {signal} \
  -clct [get_rp_groups {test1}]
```

SEE ALSO

add_to_rp_group(2)
create_rp_group(2)
get_rp_groups(2)
remove_from_rp_group(2)
remove_rp_groups(2)

gui_load_rp_vm

Loads the data for relative placement (RP) group visual mode.

SYNTAX

status **gui_load_rp_vm**
[-color group | hierarchy]
[-num_colors *count*]
[-cell_only]

Data Types

count integer

ARGUMENTS

-color group | **hierarchy**

Specifies whether to color objects by relative placement group or relative placement hierarchy level.

-num_colors *count*

Specifies the number of colors to be used in the visual mode.

-cell_only

Colors cells inside the RP group only.

DESCRIPTION

This command loads the data for relative placement (RP) visual mode. Relative placement visual mode helps to examine the placement of the instances (cells and other RP groups) in each group. This visual mode displays a colored map of RP groups with a different color either for each group (by default) or for each relative placement hierarchy level.

EXAMPLES

The following example uses 11 colors to color the cells based on relative placement groups.

```
prompt> gui_load_rp_vm -color group \
-num_colors 11 -cell_only
```

SEE ALSO

None

gui_load_scan_chain_vm

Loads the data for scan chain visual mode.

SYNTAX

status **gui_load_scan_chain_vm**
-chains *scan_chain_names_list*

Data Types

scan_chain_names_list list

ARGUMENTS

-chains *scan_chain_names_list*

Loads the list of scan chains into the visual mode.

DESCRIPTION

Loads the data for scan chain visual mode. This visual mode can be used to examine the placement of scan cells in the design after a scan is inserted. This visual mode colors each scan chain with a different color.

Note that the cell count reported by this command includes all cell objects along the scan path.

EXAMPLES

The following example displays the scan chains 1, 2, and 3.

prompt> **gui_load_scan_chain_vm -chains {1 2 3}**

SEE ALSO

None

gui_load_voltage_area_vm

Loads the data for voltage area visual mode.

SYNTAX

status **gui_load_voltage_area_vm**

Data Types

ARGUMENTS

DESCRIPTION

The command generates the visual mode view of voltage areas. Each voltage area is colored based on its type.

EXAMPLES

The following example will place all the voltage areas in the design into different bins based on their types.

prompt> **gui_load_voltage_area_vm**

SEE ALSO

None.

gui_log_cmd

Log a Tcl command without executing it

SYNTAX

string **gui_log_cmd** -cmd *command*
  [-comment | -no_comment]
  [-suppress_newline | -no_suppress_newline]

string *command*

ARGUMENTS

-cmd *command*

The command to log.

-comment

Enables commenting out of the logged command. If neither -comment nor -no_comment is specified, -no_comment is the default.

-no_comment

Disables commenting out of the logged command.

-suppress_newline

The logged command will not include an added newline character. If neither -suppress_newline nor -no_suppress_newline is specified, -no_suppress_newline is the default.

-no_suppress_newline

The logged command will include an added newline character.

DESCRIPTION

Log a Tcl command without executing it.

The return value of the command is 1.

gui_log_cmd is a thin wrapper around EkShellExecution::log(), which in turn is a thin wrapper around cci_log_command.

The -comment, -no_comment, -suppress_newline, -no_suppress_newline options are there for completeness (cci_log_command

has comment and suppress_newline parms).

The command is hidden.

EXAMPLES

```
psyn_shell-t> gui_log_cmd -cmd {echo abc}
1
```

SEE ALSO

gui_eval_cmd(2)

gui_log_performance

Controls the start and stop of a performance data logging process and outputs into a file.

SYNTAX

string **gui_log_performance**
[-start *file_name*|-stop]
[-compress]
[-quiet]

Data Types

file_name string

ARGUMENTS

-start *file_name*

This option is mutually exclusive with -stop. The argument is the path name of the file to which the log is output. The option begins logging performance data. If the file already exists, the command overwrites the existing file.

-stop

This option is mutually exclusive with -start. This option stops logging performance data. After stopped, the log cannot be reopened for appending.

-compress

If given, the log is saved as a compressed file. This option must be combined with -start.

-quiet

If given, this option causes the command to ignore error conditions.

DESCRIPTION

This command controls the start and stop of a performance data logging process. The performance data is logged before and after executing the commands which are required. The data includes cpu, memory and time. After successfully start a log, the command returns the path name with extension .log or .log.gz if not given in *file_name*.

EXAMPLES

Start logging and save the log to a file without compressing.

prompt> **gui_log_performance -start ./test**
/an/absolute/path/.test.log

Start logging and save the log to a compressed file.

prompt> **gui_log_performance -start ./test -compress**
/an/absolute/path/.test.log.gz

Stop logging

prompt> **gui_log_performance -stop**

SEE ALSO

gui_measure_charts_text

Measure text size in view or plot

SYNTAX

value **gui_measure_charts_text**
{ -view *view_name* | -plot *plot_name* }
-name *string*
-text *string*
[-html]

Data Types

view_name string
plot_name string
value string

ARGUMENTS

-view *view_name*

View to get text size from.

-plot *plot_name*

Plot to get text size from.

-name *string*

Size value to return.

The supported names are width, height, ascent and descent.

width Return the text width in view or plot units

height Return the text height in view or plot units

ascent Return the text ascent in view or plot units

descent Return the text descent in view or plot units

-text *string*

Text to measure

-html

Specifies that the text is in HTML format.

RETURNS

value

Returned text size in view or plot units.

DESCRIPTION

Get the size of text displayed in view or plot for manual placement annotations.

EXAMPLES

The following example gets the width of the string "Abc" in a plot.

```
shell> gui_measure_charts_text -plot $plot -text "Abc" -name width
```

SEE ALSO

gui_get_charts_property(2)
gui_get_charts_data(2)

gui_merge_utable

Merge 2 (loaded) tables via the given join column(s).

SYNTAX

string **gui_merge_utable**
-name *Table_Name*
-merge_table *Table_Name*
-join_columns *Column_Names*
[-output_table *Table_Name*]
[-columns *Column_Names*]

Table_Name String
Column_Names List of Strings

ARGUMENTS

-name *Table_Name*

A primary table name to merge into unless an output table name is also given.

-merge_table *Table_Name*

The merge table name. This table must have the same 'join columns' as the primary table.

-join_columns *Column_Names*

The join columns can be one or more columns that provide a unique key that only matches one row at a time. This usually is a full object name like a net,port or cell name, but can also be a 'key set' like a full_name plus object_class.

-output_table *Table_Name*

An **optional** output table name where the resultant merged table is stored, default is the primary table.

-columns *Column_Names*

An **optional** merge column list of columns that specify the columns to merge from the merge table, default is all.

DESCRIPTION

This command allows you to merge two loaded tables with a common set of

(key/join) columns and produces a resultant merged table that can be saved under a new table name or by default, will be saved in the original primary table. Both tables have to be loaded in memory, see links for gui_import_utable and gui_open_utable below to read in tables.

EXAMPLES

The following example merges two tables (tableA & tableB) joining them via the common key columns (full_name & object_class) and then outputs the result into tableC. It also only merges the column 'power' from tableB with all of tableA, by default all columns in tableB would be merged.

```
shell> # Import tableA:
    gui_import_utable -file tableA.csv
    # Import tableB:
    gui_import_utable -file tableB.csv

    # Merge them into a new tableC
    gui_merge_utable -name tableA -merge tableB -output tableC \
        -join_columns {full_name object_class} -columns {power}

    # View merged tableC:
    gui_show_utable -name tableC
```

SEE ALSO

gui_get_utable(2)
gui_import_utable(2)
gui_export_utable(2)
gui_open_utable(2)
gui_write_utable(2)

gui_mouse_tool

Operate mouse tools of a given view

SYNTAX

gui_mouse_tool
-window *window*
[-start *tool* |
-current |
-list |
-add_point *{x y}* |
-drag *{{x1 y1} {x2 y2}}* |
-delete_point |
-apply |
-reset |
-abort |
-cycle |
-cycle_back]

window *String*
tool *String*

ARGUMENTS

-window *window*

window specifies the window of which this command should operate the mouse tool.

-start *tool*

Starts the given mouse tool.

-current

Return the name of current active tool.

-list

Return the names of all available tools.

-add_point *{x y}*

Add an input point

-drag *{{x1 y1} {x2 y2}}*

Perform drag operation between two points.

-delete_point

Removes last input point

-apply

Performs tool action

-reset

Clears all pending input or ends the tool if no pending

-abort

Aborts mouse tool unconditionally

-cycle

Cycles through choices for a given operation

-cycle_back

Cycles back through choices for a given operation

DESCRIPTION

The **gui_mouse_tool** controls the mouse tool of a view that supports mouse actions log and replay capabilities. Not all views need to support all of the functionality provided by the interface of this command. User can use the command to script varios mouse action.

EXAMPLES

The following example has the layout view start the zoom in tool and perform a zoom to particular area.

```
> gui_mouse_tool -window Layout.1 -start ZOOM_IN_TOOL
> gui_mouse_tool -window Layout.1 -add_point {27.461 433.400}
> gui_mouse_tool -window Layout.1 -add_point {51.261 393.123}
> gui_mouse_tool -window Layout.1 -abort
```

SEE ALSO

gui_open_error_data

Opens an error data file in the error browser.

SYNTAX

gui_open_error_data
[-name *error_data_name*]
[-file_name *file_name*]
[*error_data*]

Data Types

error_data_name string
file_name string
error_data list

ARGUMENTS

-name *error_data_name*

Specifies the data name of the error data to open in the error browser. If other error data was already loaded into the error browser, the specified data is added to the existing data. This option is mutually exclusive with the **-file_name** and *error_data* options.

-file_name *file_name*

Specifies the name of the exported error file to open in the error browser. The name of the error data file to be opened and shown in the error browser. If other error data was already loaded into the error browser, the specified data is added to the existing data. This option is mutually exclusive with the **-name** and *error_data* options.

error_data

Specifies the data name or list of data names to open in the error browser. If the error data is already open, then a collection of the error data or the error data name can be used to show it in the error browser.

DESCRIPTION

The **gui_open_error_data** command displays the given error data in the GUI error browser. If the error browser is not already visible, it is first made visible.

If you specify a file name with the **-file_name** option, the command first opens an error data file with the given file name.

Note that if the command first opens an error data, it increments the open_count for an error data. Incrementing the open_count needs to be balanced by the same number of decrementing the open_count before the error data is closed and removed from memory.

The **gui_open_error_data** command opens an error data only when necessary. It does not increment the open_count if the error data is already open and in memory. If the **gui_open_error_data** command increments the open_count, then a subsequent call to **gui_close_error_data** for the same error data decrements the open_count. Otherwise, a subsequent call to **gui_close_error_data** simply removes the error_data from the error browser without decrementing the open_count.

EXAMPLES

The following example opens the physical DRC error data file that is named "my_design_dppinassgn.err" and shows it in the error browser.

```
prompt> gui_open_error_data -file_name my_design_dppinassgn.err]
```

The following example first opens the physical DRC error data file that is named "my_design_dppinassgn.err", then shows it in the error browser.

```
prompt> set data [open_drc_error_data -file_name my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
prompt> gui_open_error_data $data
```

SEE ALSO

close_drc_error_data(2)
collections(2)
create_drc_error_data(2)
get_drc_error_data(2)
gui_close_error_data(2)
gui_get_error_data(2)
open_drc_error_data(2)
remove_drc_error_data(2)

gui_open_utable

Open the given UserTable file.

SYNTAX

string **gui_open_utable**
-file *File_Name*
[-name *Table_Name*]
[-read_only]

File_Name String
Table_Name String

ARGUMENTS

-file *File_Name*

The name of the file to open and copy the UserTable into memory.

-name *Table_Name*

The name of the user table located in the file. By default the name of the file minus the extension is used.

-read_only

This flag stops the table from being copied into memory and instead the table contents are streamed which can save a lot of system memory if the table is huge.

DESCRIPTION

This command opens the given UserTable file and copies the table contents into memory unless the **-read_only** flag is given which causes the contents to be streamed in as needed. This can save system memory if the table is huge.

EXAMPLES

The following example opens the UserTable file 'my_table.utable' and copies the

table 'my_table' into memory.

```
shell> gui_open_utable -file my_table.utable
```

The following example opens the given UserTable file but instead of copying it into memory it connects in read-only mode, steaming the contents as needed.

```
shell> gui_open_utable -file my_table.utable -read_only
```

SEE ALSO

gui_append_utable(2)
gui_change_selection_utable(2)
gui_close_utable(2)
gui_create_utable(2)
gui_export_utable(2)
gui_import_utable(2)
gui_show_utable(2)
gui_write_utable(2)

gui_overlay_layout

Add/Remove/Adjust a design overlay to a layout window

SYNTAX

gui_overlay_layout
-window *windowID*
-design *design*
-add
-remove
-brightness *value*

windowID *String*
design *String*
add *Boolean flag*
remove *Boolean flag*
brightness *Integer*

ARGUMENTS

-window *windowID*

windowID specifies the layout window to add the overlay

-design *design*

design specifies the design to overlay

-add

add the design as an overlay

-remove

remove the design as an overlay

-brightness *value*

adjust the brightness of the overlay design

DESCRIPTION

Use this command to add, remove, or adjust an overlay on a layout view. The design to overlay must already be an opened design. You cannot overlay the same design twice nor can you overlay a design on itself. The -add and -remove flags are mutually exclusive. The brightness of an overlay can be varied from 0 (not visible at all) and 100 (fully bright). The -remove flag is mutually exclusive with the -brightness flags.

EXAMPLES

shell> **gui_overlay_layout -window Layout.1 -design fill_view -add**
shell> **gui_overlay_layout -window Layout.1 -design fill_view -brightness Off**
shell> **gui_overlay_layout -window Layout.1 -design fill_view -brightness 33%**
shell> **gui_overlay_layout -window Layout.1 -design fill_view -remove**

gui_place_charts_plots

Place plots in charts view

SYNTAX

status **gui_place_charts_plots**
-view *view_name*
[-vertical]
[-horizontal]
[-rows *int*]
[-columns *int*]
[*plots*]

Data Types

view_name string
plot_id_list list

ARGUMENTS

-view *view_name*

Charts View to place plots in.

-vertical

Stack plots vertically.

-horizontal

Stack plots horizontally.

-rows *int*

Stack plots in grid using the specified number of rows.

-columns *int*

Stack plots in grid using the specified number of columns.

plots

List of plots to place. If not specified then all plots are placed

DESCRIPTION

Place plots in a view.

EXAMPLES

The following example stacks all plots horizontally.

```
shell> gui_place_charts_plots -view $view -horizontal
```

SEE ALSO

gui_create_charts_plot(2)
gui_group_charts(2)

gui_query_objects

Get the value of a query text for a collection of object.

SYNTAX

string **gui_query_objects**
object_collection

Data Types

string *object_collection*

ARGUMENTS

object_collection

Specifies the collection of objects to use to get the query text.

DESCRIPTION

The **object_collection** command returns and displays the query text. Same text will be shown in Query toolbar for a particular object query.

A **object_collection** is collection of objects for which the query text will be generated. The collection of objects might be returned as the result of another command such as the **get_cells** command.

For information about collections, see the **collections** man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example displays query text for the all plan groups in the design.

```
prompt> gui_query_objects [get_plan_groups]
**********************************
```

```
Object summary:
plan_group : 5
**********

plan_group : RES_I109

plan_group : RES_I110

plan_group : RES_I111

plan_group : RES_I112

plan_group : RES_I113
```

The following example displays a query text for cell named RES_I112/nr02d0_B3_1.

```
prompt> gui_query_objects [get_cells RES_I112/nr02d0_B3_1]
**********
cell : RES_I112/nr02d0_B3_1
**********
allowable_orientation          N FN FS S FW W E FE
area                           25.600000
aspect_ratio                   2.500000
bbox                           {305.600 155.200} {308.800 163.200}
boundary                       {305.600 155.200} {308.800 155.200} {308.800 163.200} {305.600 163.200} {305.600 155.200}
cell_id                        3
design                         test
eco_status                     eco_reset
fp_area                        25.600000
fp_aspect_ratio                2.500000
fp_height                      8.000000
fp_number_of_hard_macro        0
fp_number_of_io_cell           0
fp_number_of_macro             0
fp_number_of_standard_cell     0
fp_width                       3.200000
full_name                      RES_I112/nr02d0_B3_1
height                         8.000000
is_black_box                   false
is_fixed                       false
is_hard_macro                  false
is_hierarchical                false
is_io                          false
is_jtag                        false
is_logical_black_box           false
is_macro_shape_fixed           false
is_mim                         false
is_mim_master_instance         false
is_physical                    true
is_physical_black_box          false
is_physical_only               false
is_placed                      true
is_preserved                   false
is_soft_fixed                  false
is_soft_macro                  false
is_spare_cell                  false
mask_layout_type               std
```

```
movebound                RES_I112
name                     nr02d0_B3_1
number_of_pins           3
object_class             cell
object_id                9782
orientation              N
origin                   305.600 155.200
outer_keepout_margin     no_outer_keepout_margin
ref_lib_name             /remote/dtdata1/testdata/designs/syn/ggui/read_cell_demo/mw/cb25_typ
ref_name                 nr02d0
ref_view_name            FRAM
sm_estimation_mode       unestimated
width                    3.200000
within_ilm               false
```

SEE ALSO

collections(2)
get_selection(2)
query_objects(2)

gui_read_timing_paths

Reads a persistent path data file containing a timing path collection from some design.

SYNTAX

string **gui_read_timing_paths**
-file *name*
[-report *type*]
[-blocks *insts*]
[-cells *clct*]
[-strict_validate]

Data Types

name string
type Values: summary, qor
insts list of block instances for a common block reference
clct collection of cell objects

ARGUMENTS

-file *name*

Specifies the name of the persistent path data file to be read.

-report *type*

Produces various summary reports on the contents of the path data file.

Note: When this option is used, it will print the report and return. No paths are loaded.

'summary': Produces a header of the contents of the file.
'qor': Produces a summary QOR report for the file.

-blocks *insts*

Given a list of block instances, read-only paths with points that are contained in these instances. The point names and objects are adjusted to the block scope shared by these instances. This allows loading a single block design and looking at paths that are in that block design. Note: This option is mutually exclusive with the **-cells** option.

-cells *clct*

Given a collection of cells in the current design, read-only paths with points that are contained in the given cell collection. Note: This option is mutually exclusive with the **-blocks** option.

-strict_validate

As the paths are read in and validated against the current design, exit on any error. Otherwise only messages are produced but paths are still created.

DESCRIPTION

The **gui_read_timing_paths** command reads a persistent path data file and re-creates the associated path collection. The persistent path data file written by **gui_write_timing_paths** contains all the information needed to re-create the timing paths.

EXAMPLES

The following examples read a timing path collection from a persistent path data file.

prompt> **set mypaths [gui_read_timing_paths -file "my.paths"]**

SEE ALSO

gui_write_timing_paths(2)

gui_remove_all_annotations

Removes specified group of annotations or all annotations from the specified layout window or from all windows.

SYNTAX

status **gui_remove_all_annotations**
[-window *window_name*]
[-group *group_name*]

Data Types

window_name string
group_name string

ARGUMENTS

-window *window_name*

Specifies the instance name of the layout window.

-group *group_name*

Specifies the annotation group name.

DESCRIPTION

This command removes specified group of annotations from the specified layout window. If group name is omitted, it will remove all annotations groups. If window name is omitted, it will remove annotations from all windows. The command returns 1 if it is successful.

EXAMPLES

```
prompt> gui_add_annotation -window Layout.1 -group Test1 \
  -type rect -text Test1 -color green -width 2 \
  -pattern Dense5Pattern {{200 200} {777 777}}
1
prompt> gui_remove_all_annotations -window Layout.1
```

1

SEE ALSO

gui_add_annotation(2)
gui_remove_annotations(2)

gui_remove_all_rulers

Removes rulers from the specified layout window or from all windows.

SYNTAX

status **gui_remove_all_rulers**
[-window *window_name*]

Data Types

window_name string

ARGUMENTS

-window *window_name*

Specifies the instance name of the layout window.

DESCRIPTION

This command removes rulers form specified layout window, or from all windows if the **-window** option is omitted, and returns 1 if it is successful.

EXAMPLES

prompt> **gui_remove_all_rulers -window Layout.1**
1

SEE ALSO

gui_remove_ruler(2)

gui_remove_annotations

Removes specified group of annotations from the specified layout window.

SYNTAX

status **gui_remove_annotations**
[-window *window_name*]
[-group *group_name*]
[*anno*]

Data Types

window_name string
group_name string
anno collection

ARGUMENTS

-window *window_name*

Specifies the instance name of the layout window.

-group *group_name*

Specifies the annotation group name.

anno

Specifies a collection of annotations to remove. Get a collection of annotations with the gui_get_annotations command. This option cannot be specified with any other option.

DESCRIPTION

This command removes specified group of annotations from the specified layout window. If group name is omitted, global group name is used. If window name is omitted, global window name is used. The command returns 1 if it is successful.

EXAMPLES

```
prompt> gui_add_annotation -window Layout.1 -group Test1 \
  -type rect -text Test1 -color green -width 2 \
  -pattern Dense5Pattern {{200 200} {777 777}}
1
prompt> gui_remove_annotations -window Layout.1
1
prompt> gui_remove_annotations [gui_get_annotations -filter client_data==MyData]
1
```

SEE ALSO

gui_add_annotation(2)
gui_remove_all_annotations(2)
gui_get_annotations(2)

gui_remove_category_nodes

Removes the offspring category nodes and any category node sub-trees rooted at the specified parent category nodes.

SYNTAX

gui_remove_category_nodes
[-tree *category_tree*]
-parent_categories *category_nodes_list*

Data Types

category_tree	string or category_tree collection
category_nodes_list	list

ARGUMENTS

-tree *category_tree*

The category tree in which the category node operation will happen; must be a category tree name or a category tree collection of size one; the most recently used category tree is used if the *-tree* option is omitted.

-parent_categories *category_nodes_list*

List of hierarchical (parent category node) name patterns and collections of parent category nodes under which one or more offspring category nodes will be removed. A parent category node name is a unique hierarchical name of a category node in a Tcl list format.

DESCRIPTION

Removes the offspring category nodes and any category node sub-trees rooted at the specified parent category nodes. All direct and indirect offspring category nodes of the specified parent category nodes are removed.

The command returns "1" if successful. Otherwise, an empty string is returned.

You should use this command only if at least one category tree is available. The command acts on the category tree specified.

EXAMPLES

In the following example, all category nodes that are direct or indirect children of the ALL root category node are removed. Only the ALL root category node remains.

```
shell> gui_remove_category_nodes -parent_categories {ALL}
1
```

In the following example, the offspring of the {Launch Clock: CLK1} category node is removed. This category node is a child of the {Pathgroup: CLK1} category node, which in turn is a child of the ALL root category node.

```
shell> gui_remove_category_nodes -parent_categories \
?   {{ALL {Pathgroup: CLK1} {Launch Clock: CLK1}}}
1
```

SEE ALSO

gui_create_category_nodes(2)
gui_get_category_nodes(2)

gui_remove_category_rules

Removes all category rules except built-in rules by default.

SYNTAX

string **gui_remove_category_rules**
[-all | -names *rule_names_list*]

Data Types

rule_names_list list

ARGUMENTS

-all

Removes all category rules, including built-in rules.

This option and the **-names** option are mutually exclusive. If you do not specify either of these options, the command removes all category rules except the built-in rules.

-names *rule_names_list*

Specifies the names of the category rules to be removed.

This option and the **-all** option are mutually exclusive. If you do not specify either of these options, the command removes all category rules except the built-in rules.

DESCRIPTION

This command removes category rules that have already been created by the **gui_create_category_rule** command during the current run of the tool.

When you use this command without specifying any options, it removes all category rules except built-in rules. Specify the **-all** option to remove all category rules, including the built-in rules. Specify the **-names** option to remove individual rules by name.

The command returns an empty string as its result.

EXAMPLES

The following example removes all category rules, including the built-in rules:

```
prompt> gui_remove_category_rules -all
```

The following example removes all category rules except the built-in rules:

```
prompt> gui_remove_category_rules
```

The following example removes the rules named Rule1 and Rule2:

```
prompt> gui_remove_category_rules -names {Rule1 Rule2}
```

The following example removes non-built-in rules at the top of a user-defined category rule creation script, which is being developed interactively. You can repeatedly change and refine this script by using a text editor, and you can source the script multiple times in a single run of the tool.

```
# first remove all existing user-defined non-built-in rules
gui_remove_category_rules
# now create the user-defined category rules
gui_create_category_rule ...
gui_create_category_rule ...
and so forth
```

SEE ALSO

gui_create_category_rule(2)
gui_list_category_rules(2)

gui_remove_category_trees

Removes the specified category trees (or all category trees by default).

SYNTAX

string **gui_remove_category_trees**
[*category_trees_list*]

Data Types

string	the string data type
category_trees_list	list (of category tree name patterns and collections)

ARGUMENTS

category_trees_list

A list of of category tree name patterns and category tree collections specifying the category trees to be removed. If this option is omitted, then all existing category trees are removed.

Each category tree name pattern must be either the exact name of an existing tree, or the pattern "*", which means "all category trees". More general forms of wild-card matching are not supported at this time.

DESCRIPTION

Removes the specified category trees (or all category trees by default). An empty string is returned.

EXAMPLES

This example show the creation of a couple of category trees, and the results of some different runs of gui_remove_category_trees:

```
shell> set path_clct [get_timing_paths ...]
_sel18
shell> set path_clct2 [get_timing_paths ...]
_sel19
shell> gui_create_category_tree -name tree0 -collections $path_clct
_sel20
```

```
shell> gui_create_category_tree -name tree1 -collections $path_clct2
_sel21
shell> gui_remove_category_trees tree1
shell> gui_remove_category_trees *
```

SEE ALSO

gui_create_category_tree(2)
gui_get_category_trees(2)

gui_remove_cell_block_marks

Removes the block mark string values from one or more cells.

SYNTAX

status **gui_remove_cell_block_marks**
-all | *cells*

Data Types

cells list

ARGUMENTS

-all

Removes all block marks on all cells that are marked.

This option and the *cells* option are mutually exclusive, and one of them must be specified.

cells

Removes block marks from one or more cells specified in a list of cell name patterns or cell collections.

This option and the **-all** option are mutually exclusive, and one of them must be specified.

DESCRIPTION

This command removes the block mark string values from one or more hierarchical or leaf-level cell instances. If the **-all** option is specified, then all the block marks are removed from all the cell instances that are marked. If the *cells* option is specified, then block marks are removed only from the specified cell instances.

EXAMPLES

The following example sets the block mark for a hierarchical cell instance identified by a cell name, and then removes the block mark from that cell instance:

prompt> **gui_set_cell_block_marks I_TOP/I_ALU ALU**

```
prompt> gui_remove_cell_block_marks I_TOP/I_ALU
```

The following example sets the block mark for a hierarchical cell instance identified by a nested run of the **get_cells** command, and then removes the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks [get_cells I_TOP/I_ALU]  ALU
prompt> gui_remove_cell_block_marks [get_cells I_TOP/I_ALU]
```

The following example removes all the block marks from all the cell instances that are marked:

```
prompt> gui_remove_cell_block_marks -all
```

SEE ALSO

gui_set_cell_block_marks(2)
gui_get_cell_block_marks(2)
gui_list_cell_block_marks(2)

gui_remove_charts_annotation

Remove annotation object from view or plot

SYNTAX

status **gui_remove_charts_annotation**
{ -view *view_name* | -plot *chart_id* }
{ -id *string* | -all }

Data Types

view_name string
chart_id string

ARGUMENTS

-view *view_name*

View to remove annotations from.

-plot *chart_id*

Plot to remove annotations from.

-id *string*

Id of annotation to remove.

-all

Remove all remove annotations.

DESCRIPTION

Removes one or all annotations from a chart or a view.

EXAMPLES

The following example removes all annotations from a view.

```
shell> gui_remove_charts_annotation -view $view -all
```

SEE ALSO

gui_create_charts_arrow_annotation(2)
gui_create_charts_ellipse_annotation(2)
gui_create_charts_point_annotation(2)
gui_create_charts_polygon_annotation(2)
gui_create_charts_polyline_annotation(2)
gui_create_charts_rectangle_annotation(2)
gui_create_charts_text_annotation(2)

gui_remove_charts_model

Remove model from charts

SYNTAX

status **gui_remove_charts_model**
-model *model_id*

Data Types

model_id int

ARGUMENTS

-model *model_id*

Charts model to remove.

DESCRIPTION

Remove model data from charts.

Note: It is recommended that existing plots using the model should also be removed as they may lose access to some or all of the original data.

EXAMPLES

The following example removes a model.

shell> gui_remove_charts_model -model $model

SEE ALSO

gui_create_charts_model(2)

gui_remove_charts_plot

Remove one or all plots from a charts view

SYNTAX

status **gui_remove_charts_plot**
-view *view_name*
{ -plot *chart_id* | -all }

Data Types

view_name string
chart_id string

ARGUMENTS

-view *view_name*

View to remove chart from.

-plot *chart_id*

Chart to remove.

-all

Remove all changes.

DESCRIPTION

Removes one or all charts from a view.

EXAMPLES

The following example removes all charts from a view.

shell> gui_remove_charts_plot -view $view -all

SEE ALSO

gui_create_chart(2)

gui_remove_message_waivers

Removes message waivers created in the Message Browser or by using the **gui_create_message_waiver** command.

SYNTAX

status **gui_remove_message_waivers**
-name *waivers_name* | -all

Data Types

waivers_name string

ARGUMENTS

-name *waivers_name*

Specifies the name of the waivers to delete. This option is mutually exclusive with the **-all** option.

-all

Remove all waivers. This option is mutually exclusive with the **-name** option.

DESCRIPTION

The **gui_remove_message_waivers** command removes message waivers created in the Message Browser or by running the **gui_create_message_waiver** command.

EXAMPLES

The following example removes waiver 'mywaiver':

prompt> **gui_remove_message_waivers -name mywaiver**

SEE ALSO

gui_create_message_waiver(2)

gui_remove_pref_key

Remove a preference key.

SYNTAX

string **gui_remove_pref_key** -key *Key*
[-category *Category*]

Key *String*
Category *String*

ARGUMENTS

-key *Key*

Key specifies the key which will be used together with the specified category to uniquely identify the key to be removed.

-category *Category*

Category specifies the category that owns the specified key. If not specified, a default category will be used.

DESCRIPTION

This command removes the preference key specified by the key name.

SEE ALSO

gui_create_pref_category(2)
gui_create_pref_key(2)
gui_set_pref_value(2)
gui_get_pref_value(2)
gui_get_pref_value_type(2)
gui_get_pref_categories(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_remove_ruler

Removes the specified rulers.

SYNTAX

status **gui_remove_ruler**
[-window *window_name*]
-point *point* | -all

Data Types

window_name string
point point

ARGUMENTS

-window *window_name*

Specifies the instance name of the layout window.

-point *point*

Specifies the coordinates of the point near the ruler to be removed, in following format:

{x y}

-all

Removes all rulers.

DESCRIPTION

This command removes rulers from the specified layout window, or from all windows if the **-window** option is omitted, and returns 1 if it is successful.

EXAMPLES

prompt> **gui_remove_ruler -window Layout.1 -point {400 400}**

```
1
prompt> gui_remove_ruler -window Layout.1 -all
1
```

SEE ALSO

gui_remove_all_rulers(2)

gui_remove_var

Remove the variable with the specified name.

SYNTAX

void **gui_remove_var** -name *VarName*

VarName *String*

ARGUMENTS

-name *VarName*

VarName specifies the name of the gui variable.

DESCRIPTION

This command removes the gui variable with the specified name. This variable is created with the *gui_create_var* command.

SEE ALSO

gui_create_var(2)
gui_set_var(2)
gui_get_var(2)
gui_exist_var(2)

gui_remove_vm

Remove Visual Mode

SYNTAX

string **gui_remove_vm** -name *identifier*

string *identfier*

ARGUMENTS

-name *identifier*

Specifies the name of the visual mode to be removed.

DESCRIPTION

Removes the specified visual mode and all the associated buckets.

EXAMPLES

To remove visual mode named "mycoloring1", use the following command:

shell> **gui_remove_vm -name mycoloring1**

SEE ALSO

gui_create_vm(2)
gui_create_vmbucket(2)
gui_get_vm(2)
gui_get_vmbucket(2)
gui_list_vm(2)
gui_remove_vmbucket(2)
gui_set_vmbucket(2)

gui_update_vm(2)

gui_remove_vmbucket

Removes the specified bucket or all buckets from the specified visual mode

SYNTAX

string **gui_remove_vmbucket**
-vmname *mode_identifier*
[-name *bucket_identifier*]
[-all]

Data Types

mode_identifier string
bucket_identifier string

ARGUMENTS

-vmname *mode_identifier*

Specifies the visual mode of which the bucket is a member. The visual mode must already exist.

-name *bucket_identifier*

Specifies the name of the bucket to be removed from the specified visual mode.

-all

Removes all buckets associated with the specified visual mode.

DESCRIPTION

Removes the specified bucket or all buckets from the specified visual mode.

EXAMPLES

To remove the visual mode bucket named red from the visual mode named vm1, enter the following command:

prompt> **gui_remove_vmbucket -vmname vm1 -name red**

To remove all the visual mode buckets from the visual mode named vm1, enter the following command:

prompt> **gui_remove_vmbucket -vmname vm1 -all**

SEE ALSO

gui_create_vm(2)
gui_create_vmbucket(2)
gui_get_vm(2)
gui_get_vmbucket(2)
gui_list_vm(2)
gui_remove_vm(2)
gui_set_vmbucket(2)
gui_update_vm(2)

gui_report_errors

Output a textual report of errors.

SYNTAX

string **gui_report_errors**
  [-errors *Errors*]
  [-file *FileName*]
  [-include_hidden]

Errors *String*
FileName *String*

ARGUMENTS

-errors *Errors*

The **-errors** specifies the handle for a collection of errors to report on. This argument is optional. If omitted, a report is output for the errors currently loaded in the Error Browser error list.

-file *FileName*

FileName specifies the name of the file to write the report to. This argument is optional. If omitted, the report is output to stdout.

-include_hidden

-include_hidden specifies that errors that are currently hidden from the errors list due to filters or fixed errors being hidden should be included in the report. This option is ignored if errors are given with the *-errors* option.

DESCRIPTION

This command outputs a tabular textual representation of errors. If the -errors argument is omitted and an output file is specified, performs the same operation as the Error Browser option menu command to save to file.

EXAMPLES

The following example saves a textual report of current errors to a file named myErrors.txt:

```
gui_report_errors -file "myErrors.txt"
```

SEE ALSO

gui_error_browser(2)
gui_set_current_errors(2)

gui_report_hotkeys

Report on the current hotkey bindings

SYNTAX

string **gui_report_hotkeys**
[-window_type *WindowTypeName*]

WindowTypeName *String*

ARGUMENTS

-window_type *WindowTypeName*

WindowTypeName specifies the type of top level window that the hotkey should apply to. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

DESCRIPTION

This command prints a report on the current key bindings. The list of bindings printed in the report are sorted by hotkey.

EXAMPLES

The following example creates a menu item and add an additional hotkey to it.

```
gui> gui_report_hotkeys

****************************************
Report : hotkeys
Window : LayoutWindow
Version: V-2004.06-GALILEO-BETA1-1
Date   : Tue Aug 31 10:37:17 2004
****************************************

Hot Key    Type  Function
-----------------------------------------------------
```

```
Ctrl+O    Menu  File->Open Design...
Ctrl+S    Menu  File->Save Design
Ctrl+R    Menu  Edit->Properties
M         Menu  Edit->Move...
C         Menu  Edit->Copy...
Ctrl+F    Menu  View->Zoom->Zoom Full
F         Menu  View->Zoom->Zoom Full
P         Tcl   query_objects [get_selection]
 ...
```

SEE ALSO

gui_set_hotkey(2)
gui_create_menu(2)
gui_delete_menu(2)
gui_create_toolbar(2)
gui_delete_toolbar(2)
gui_create_toolbar_item(2)
gui_delete_toolbar_item(2)
gui_default_window_type(3)
gui_custom_setup_files(3)

gui_report_map

Report information for a specified map mode.

SYNTAX

string **gui_report_map**
  -name identifier
  -report_type outputType

Data Types

identifier string

outputType string (Values: histogram, count)

ARGUMENTS

-name *identifier*

Specifies the name of the map mode to be queried.

-report_type outputType

Specifies the mode of report. Value of *outputType* can be **histogram** or **count**.

DESCRIPTION

The **gui_report_map** command queries the option values of an existing map mode name and a report mode. Histogram mode will print '**###**' as a graph bar for each bucket. Count mode will generate only count numbers of blocks in each bucket.

EXAMPLES

To generate a histogram data for the map mode named "cellDensityMap". Use the following command:

```
prompt>gui_report_map -name cellDensityMap -report_type histogram
****************************************
Report : map
Design : ORCA
```

```
Mode   : histogram
Version: P-2019.03-DEV
Date   : Tue Nov  6 22:01:35 2018
****************************************
[1.00 - 1.10)  ( 0 )
[0.90 - 1.00)  ( 0 )
[0.80 - 0.90)  ( 0 )
[0.70 - 0.80)  ( 0 )
[0.60 - 0.70)  ( 0 )
[0.50 - 0.60)  ( 0 )
[0.40 - 0.50) ############### ( 269 )
[0.30 - 0.40) ########################################################################## ( 1200 )
[0.20 - 0.30) ##################### ( 379 )
[0.00 - 0.20) ###### ( 104 )
```

SEE ALSO

gui_get_map(2)
gui_get_mapbucket(2)

gui_report_performance

Generates a performance report.

SYNTAX

status **gui_report_performance**
[-type *report_type*]
[-file *file_name*]
[-compress *compress_method*]

Data Types

report_type string
file_name string
compress_method string

ARGUMENTS

-type *report_type*

Specifies the report type. If report type is omitted or the value of this option is simple, output a simple version report. If the value of this option is detail, output a detail version report which includes additional information of x11 performance, region searching, attribute querying and render time.

The supported values for this option are

simple
detail

-file *file_name*

Outputs the report as a file and specifies the file name. If file name is omitted, output the report to console.

-compress *compress_method*

Specifies the compress method. The supported values for this option are

none
gzip

DESCRIPTION

This command report the performance data of machine state, design information, GUI configurations and current GUI views state.

EXAMPLES

Generate a simple version report and output to console.

prompt> **gui_report_performance**

Generate a detail version report and output to a file.

prompt> **gui_report_performance -type detail -file file_name**

Generate a detail version report and output to a compressed file

prompt> **gui_report_performance -type detail -file file_name -compress gzip**

SEE ALSO

gui_report_proc_arg_type_names

Report on the available type_name attribute values.

SYNTAX

string **gui_report_proc_arg_type_names**

DESCRIPTION

This command prints a report on the available values for the type_name attribute for the **-define_args** argument to the **define_proc_attributes** command.

When a tcl proc has been defined using the **define_proc_attribute** command, a GUI form for the command will be auto-generated and shown from the application command search feature or using the command **gui_show_command_form**. The kinds of proc argument input fields used in the generated GUI form depend on the **type_name** attribute values provided for command options. The **type_name** attribute value for an option is given as a part of the **-define_args** argument. The **-define_args** option accepts a list of the following values, in this order:

* option name * option decription * option value type description * data_type * list of option attribute name-value pairs

Here is the usage for the **type_name** attribute within argument value for the **define_proc_attributes** command option **-define_args**:

```
define_proc_attributes <procedure name> \
  -define_args { \
    {<option name> <option description> <value type description> <data_type> \
      {[required | optional] {type_name <type_name_value>}} \
  } \
}
```

where <type_name_value> can be substituted with any of the type names reported by **gui_report_proc_arg_type_names**.

For the **type_name** attribute value to have effect on the input field used in a GUI form, the <data_type> value must be **string**, for a single value, or **list** for multiple values. The value **list** is not supported for all **type_name** values as shown in the report's **data_type** column.

The following is an example of a **define_proc_attribute** command invocation with options accepting a collection or names of nets:

```
define_proc_attributes myProc \
  -define_args { \
    {-net "select a net" "net name or a collection of a net" string \
        {optional {type_name net}}} \
    {-nets "select nets" "net names or a collection of nets" list   \
        {optional {type_name net}} \
  } \
}
```

In the above example, the **-net** option has **data_type** value **string** and **type_name** value **net**. This option accepts a single net as input. The **-nets** option has **data_type** value **list** and **type_name** value **net**. This option accepts multiple nets as input.

The report produced by this command lists the available **type_name** values. When one of these supported **type_name** values is used, the GUI form for the proc will use an input field that is appropriate for user input of that kind of value. For example, for a **net** type input, as shown in the example above, the form will use an object selector field which is configured to accept net input.

A proc argument with any combination of unspecified or unsupported **data_type** and **type_name** values is represented with a simple text input field in the generated GUI form.

Some **type_name** values use input editors that allow further configuration. For example, an editor for a file name can be configured to allow input of only existing files. Or an editor for a layer name can be configured to allow input of only certain types of layers.

Available configuration options are shown in the options column, then described in more detail at the end of the report output in a separate table keyed by associate type_name values. When providing optional configuration, list the configuration name-value pairs after the type_name value as a part of the the type_name attribute value. The following is an example of a "file" type_name attribute with multiple optional configuration settings:

```
define_proc_attributes myProc \
 -define_args { \
  {-file_option "Select an existing file name" "file name"  string \
   {required
    {type_name {file {subtype exist} \
                {operation "Save As"} \
                {filter {"Text files (*.txt)" "All files (*)"}} \
           } \
    } \
   } \
  } \
}
```

EXAMPLES

The following example outputs a report of available type_name attribute values.

```
gui> gui_report_proc_arg_type_names
*****************************************************************************
Report : Supported Values for "type_name" Option Attribute
Version: L-2016.03-SP5-BETA
Date   : Fri Sep 16 08:09:34 2016
*****************************************************************************

type_name              data_type      description     options
-----------------------------------------------------------------------------
PathGroup              string, list    object type
bbox                   string, list    region
block                  string, list    object type
block_via_def          string, list    via_def
bool                   string          bool
boolean                string          boolean
bound                  string, list    object type
bundle                 string, list    object type
cell                   string, list    object type
clock                  string, list    object type
```

```
collection          string      collection
corner              string, list  object type
cut_pattern         string      cut_pattern
directory           string      directory name  context, filter,
                                    operation, subtype
 ...


Configuration Options:

type_name   option     description
------------------------------------------------------------------------
layer       subtype    Layer type -- valid values: routing, contact,
                       cut, poly, poly_cont, user, system.  Multiple
                       values may be given in a tcl list.

directory,  context    A valid directory name for current directory
file                   context for the file dialog
            filter     File or directory name filter strings to
                       set for the file dialog.  Valid value is
                       a tcl list of filter strings, e.g. {"Text
                       files (*.txt)" "All files (*)"}
            operation  Operation name for file dialog titlebar
                       and button, e.g. "Save As"
            subtype    File type: valid values are: "existing"
                       (existing file only), "any" (any file)
```

SEE ALSO

define_proc_attributes(2)
gui_show_command_form(2)

gui_report_task

task.

SYNTAX

string **gui_report_task**
  -task | -item_root *TaskName | ItemRootName*
  [-file] *FileName*

TaskName *String*
ItemRootName *String*
FileName *String*

ARGUMENTS

-task *TaskName*

TaskName specifies the name of a task on which to report. This option is mutually exclusive with the -item_root option.

-item_root *ItemRootName*

ItemRootName specifies the name of a task item root name on which to report. This option is mutually exclusive with the -task option.

-file *FileName*

FileName optionally specfies the output file into which the report is written.

DESCRIPTION

This command outputs a textual report of the given task to the xterm console. If -file is given, then the report is also output to the given file. The report will have the following approximate format. The strings bracketed with the angle brackets will be replaced with the actual attribute value.

Attributes are omitted if they are not defined. Item are shown one to a line with the hierarchy structure indicated by indentations and bars. The leaf-leveltask item names are followed by the associated task page name:

Task: <task name>
 Default: <true | false>
 Items: <items>

SEE ALSO

gui_create_task(2)
gui_get_task_list(2)

gui_schematic_add_logic

Adds logic to a schematic

SYNTAX

string **gui_schematic_add_logic** [-window *<win>*]
[-new]
[-schematic *<schem>*]
objs

string *<win>*
string *<schem>*
string *objs*

ARGUMENTS

-window *<win>*

Top level window name to place new schematic (default: current window). This option is ignored unless -new is specified.

-new

Create new schematic.

-schematic *<schem>*

Schematic view to update (default: most recently active schematic).

objs

Objects to add to the schematic.

DESCRIPTION

This command adds logic into a Path Schematic view. The command either creates a new view containing the specified objects or updates an existing view. The command always returns the name of the schematic that was updated or created.

EXAMPLES

The following example creates a new path schematic containing all the cells in the current design:

```
gui_schematic_add_logic -new [get_cells *]
```

SEE ALSO

gui_get_current_window(2)
gui_schematic_remove_logic(2)

gui_schematic_remove_logic

Removes logic from a schematic

SYNTAX

string **gui_schematic_remove_logic** [-schematic *<schem>*]
objs

string *<schem>*
string *objs*

ARGUMENTS

-schematic *<schem>*

Schematic view to update (default: most recently active schematic).

objs

Objects to remove from the schematic.

DESCRIPTION

This command removes logic into a Path Schematic view. The command returns the name of the schematic that was updated

EXAMPLES

The following example removes the cell "U1" from the most recently active schematic view:

```
gui_schematic_remove_logic [get_cells U1]
```

SEE ALSO

gui_get_current_window(2)
gui_schematic_add_logic(2)

gui_scroll

Scroll a window's viewport.

SYNTAX

gui_scroll
-window *window*
[-selection | -clct *clct* | [-habs *absX* | -hrel *relX*] [-vabs *absY* | -vrel *relY*]]

window *String*
absX *Float*
relX *Float*
absY *Float*
relY *Float*

ARGUMENTS

-window *window*

window specifies the window to be scrolled.

-selection

Determine the bounding box for all selected objects present in the view, and scroll such that the center of that box is visible.

-clct *clct*

Determine the bounding box for the collection objects present in the view, and scroll such that the center of that box is visible.

-habs *absX*

Scroll such that the specified X model position is centered in the viewport.

-hrel *relX*

Scroll horizontally by the specified number of pages. A page is the width of the viewport. Negative values scroll left and positive values scroll right.

-vabs *absY*

Scroll such that the specified Y model position is centered in the viewport.

-vrel *relY*

Scroll vertically by the specified number of pages. A page is the height of the viewport. Negative values scroll up and positive values scroll down.

DESCRIPTION

The **gui_scroll** command adjusts the position of views that support zooming. The viewport can be adjusted in the X and/or Y direction. Absolute values are expressed in model coordinates. Relative values allow paging. For example, *-hrel 1.0* means page to the right by the width of the viewport.

EXAMPLES

Scroll such that model coordinates (287.0, 1422.0) are centered in the viewport.

```
gui_scroll -window Layout.1 -habs 287.0 -vabs 1422.0
```

Scroll to the right by one viewport width.

```
gui_scroll -window Layout.1 -hrel 1.0
```

SEE ALSO

gui_zoom(2)

gui_select_bounds_of_selected

Select the bounds of the selected objects.

SYNTAX

status **gui_select_bounds_of_selected**

DESCRIPTION

The **gui_select_bounds_of_selected** command selects the bounds related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Bounds that are already selected remain selected, while bound_shape, cell, pin, port objects will have their related bounds selected.

EXAMPLES

Select the bounds of the selected objects.

```
prompt> gui_select_bounds_of_selected
```

SEE ALSO

change_selection(2)
get_bounds(2)

gui_select_bundles_of_selected

Select the bundles of the selected objects.

SYNTAX

status **gui_select_bundles_of_selected**

DESCRIPTION

The **gui_select_bundles_of_selected** command selects the bundles related to the currently selected objects. It is intended for interactive usage in the GUI for the application. This command will select the bundles associated with the nets of the selected net, routing_corridor, routing_corridor_shape, pin, port, terminal, shape, and via objects.

EXAMPLES

Select the bundles of the selected objects.

```
prompt> gui_select_bundles_of_selected
```

SEE ALSO

change_selection(2)
get_bundles(2)

gui_select_by_name

Select or highlight objects by name

SYNTAX

status **gui_select_by_name**
[{ -select | -highlight }]
-object_type *string*
[-pattern *string*]
[-filter *string*]
[{ -replace | -add | -remove }]
[-regexp]
[-nocase]
[-exact]
[-all]

ARGUMENTS

-select

Specifies that the matching objects are to be selected.

Note: only one of *-select* or *-highlight* options should be specified. If neither is specified, *-select* is assumed.

-highlight

Specifies that the matching objects are to be highlighted.

Note: only one of *-select* or *-highlight* options should be specified. If neither is specified, *-select* is assumed.

-object_type *string*

Specifies the type of objects to be selected.

The available types are dependant on the individual product. Please see your product's documentation for details.

-pattern *string*

Specifies the pattern to be used for a match.

By default matching is case-sensitive and uses wildcard patterns. If non case-sensitive matching is required use the *-nocase* option. If regular expression matching is required use the *-regexp* option. If no pattern matching is required use the *-exact* option.

-filter *string*

Specifies the expression to use when filtering the results.

-replace

Specifies that matching objects should replace the current selection or highlight.

Note: only one of the *-replace*, *-add* or *-remove* options can be specified. If none of these options area specified, then replace is assumed.

-add

Specifies that matching objects should be added to the current selection or highlight.

Note: only one of the *-replace*, *-add* or *-remove* options can be specified. If none of these options area specified, then replace is assumed.

-remove

Specifies that matching objects should be removed from the current selection or highlight.

Note: only one of the *-replace*, *-add* or *-remove* options can be specified. If none of these options area specified, then replace is assumed.

-regexp

Specifies the pattern matching should use regular expressions.

-nocase

Specifies the pattern matching should be case-insensitive.

-exact

Specifies that no pattern matching should be used so wildcards and regular expressions are ignored.

-all

Specifies that physical only objects be include in the search.

DESCRIPTION

This command allows objects of a specified type to be selected or highlighted.

It is used by the Select By Name Toolbar and Select By Name Dialog for correct log and replay of the resultant selection or highlight.

The command returns whether the search was successful or not.

EXAMPLES

The following example will add all nets matching the pattern 'A*' to the current selection.

```
shell> gui_select_by_name -select -object_type {Nets} -pattern {A*} -add
```

The following example will only highlight the Port '*Logic0*'.

```
shell> gui_select_by_name -highlight -object_type {Ports} -pattern {*Logic0*} -exact
```

SEE ALSO

change_selection(2)
gui_change_highlight(2)

gui_select_cells_of_selected

Select the cells of the selected objects.

SYNTAX

status **gui_select_cells_of_selected**

DESCRIPTION

The **gui_select_cells_of_selected** command selects the cells related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Cells that are already selected remain selected, while net, pin, shape, and via objects will have their related cells selected.

EXAMPLES

Select the cells of the selected objects.

```
prompt> gui_select_cells_of_selected
```

SEE ALSO

change_selection(2)
get_cells(2)

gui_select_connected_net_shapes

Selects the net shapes physically connected to the selected net shapes on the same nets.

SYNTAX

status **gui_select_connected_net_shapes**
[-hierarchical]
[-cross_net]

ARGUMENTS

-hierarchical

Traces connections across hierarchical module boundaries. For example, all net shapes of a connected feedthrough net are selected in one operation when this option is specified.

-cross_net

Traces physically connected shapes across different nets, except PG nets.

DESCRIPTION

The **gui_select_connected_net_shapes** command selects the physically connected net shapes of the currently selected net shapes. It is intended for interactive usage in the GUI for the application.

EXAMPLES

Select the objects of the selected net shapes.

prompt> **gui_select_connected_net_shapes**

SEE ALSO

change_selection(2)

get_shapes(2)

gui_select_connected_rdl_net_shapes

Selects the RDL net shapes physically connected to the selected RDL net shapes.

SYNTAX

status **gui_select_connected_rdl_net_shapes**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **gui_select_connected_rdl_net_shapes** command selects the physically connected RDL net shapes of the currently selected net shapes. It is intended for interactive usage in the GUI for the application.

RDL net shapes can have multiple paths from a driver to a bump. This command only selects the RDL net shapes on the same path as the currently selected net shapes.

EXAMPLES

Select the objects of the selected RDL net shapes.

```
prompt> gui_select_connected_rdl_net_shapes
```

SEE ALSO

change_selection(2)
get_shapes(2)

gui_select_connections_of_selected

Select the connections of the selected objects.

SYNTAX

status **gui_select_connections_of_selected**

DESCRIPTION

The **gui_select_connections_of_selected** command selects the connections related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Ports and Pins that are already selected remain selected, while cell, net, shape, via, terminal, bound, pin_guide, pin_blockage, and design objects will have their related connections selected.

EXAMPLES

Select the connections of the selected objects.

```
prompt> gui_select_connections_of_selected
```

SEE ALSO

gui_select_output_connections_of_selected(2)
gui_select_input_connections_of_selected(2)
change_selection(2)
get_ports(2)
get_pins(2)

gui_select_constraint_groups_of_selected

Select the constraint groups of the selected objects.

SYNTAX

status **gui_select_constraint_groups_of_selected**

DESCRIPTION

The **gui_select_constraint_groups_of_selected** command selects the constraint groups related to the currently selected nets, topology nodes, or topology edges. It is intended for interactive usage in the GUI for the application. Net, topology node, and topology edge objects will have their related constraint groups selected.

EXAMPLES

Select the constraint groups of the selected objects.

```
prompt> gui_select_constraint_groups_of_selected
```

SEE ALSO

gui_select_nets_of_selected(2)
gui_select_topology_of_selected(2)
change_selection(2)
get_constraint_groups(2)

gui_select_input_connections_of_selected

Select the input connections of the selected objects.

SYNTAX

status **gui_select_input_connections_of_selected**

DESCRIPTION

The **gui_select_input_connections_of_selected** command selects the input connections related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Ports and Pins that are already selected remain selected, while cell, net, shape, via, terminal, bound, pin_guide, pin_blockage, and design objects will have their related input connections selected.

EXAMPLES

Select the input connections of the selected objects.

```
prompt> gui_select_input_connections_of_selected
```

SEE ALSO

gui_select_output_connections_of_selected(2)
gui_select_connections_of_selected(2)
change_selection(2)
get_ports(2)
get_pins(2)

gui_select_macros_of_selected

Select the macros of the selected objects.

SYNTAX

status **gui_select_macros_of_selected**

DESCRIPTION

The **gui_select_macros_of_selected** command selects the macros within the currently selected objects. It is intended for interactive usage in the GUI for the application. Macros that are already selected remain selected, while the macros within selected design or hierarchy boundaries are also selected.

EXAMPLES

Select the macros of the selected objects.

```
prompt> gui_select_macros_of_selected
```

SEE ALSO

change_selection(2)
get_cells(2)

gui_select_matching_types_of_selected

Select the matching types of the selected cells, pins, or terminals.

SYNTAX

status **gui_select_matching_types_of_selected**

DESCRIPTION

The **gui_select_matching_types_of_selected** command selects the matching types related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Matching types that are already selected remain selected, while pins, terminals and cells will have their related matching types selected.

EXAMPLES

Select the matching types of the selected objects.

```
prompt> gui_select_matching_types_of_selected
```

SEE ALSO

change_selection(2)
get_matching_types(2)

gui_select_mib_cells_of_selected

Selects the associated MIB cells of the selected macro cells.

SYNTAX

status **gui_select_mib_cells_of_selected**

DESCRIPTION

The **gui_select_mib_cells_of_selected** command selects the associated multiply instantiated block (MIB) cells of the currently selected macro cells.

It is intended for interactive usage in the GUI for the application.

If the command fails then the selection is not changed.

EXAMPLES

Select the MIB cells of the selected macro cells.

```
prompt> gui_select_mib_cells_of_selected
```

SEE ALSO

change_selection(2)
gui_select_mib_connections_of_selected(2)

gui_select_mib_connections_of_selected

Selects the associated MIB connections of the selected macro connections.

SYNTAX

status **gui_select_mib_connections_of_selected**

DESCRIPTION

The **gui_select_mib_connections_of_selected** command selects the associated multiply instantiated block (MIB) connections (pins) of the currently selected macro connections (macro pins).

It is intended for interactive usage in the GUI for the application.

If the command fails then the selection is not changed.

EXAMPLES

Select the MIB connections of the selected macro connections.

```
prompt> gui_select_mib_connections_of_selected
```

SEE ALSO

change_selection(2)
gui_select_mib_cells_of_selected(2)

gui_select_net_buses_of_selected

Select the net buses of the selected objects.

SYNTAX

status **gui_select_net_buses_of_selected**

DESCRIPTION

The **gui_select_net_buses_of_selected** command selects the net buses related to the currently selected objects. It is intended for interactive usage in the GUI for the application. This command will select the net buses associated with the selected net.

EXAMPLES

Select the net buses of the selected objects.

```
prompt> gui_select_net_buses_of_selected
```

SEE ALSO

change_selection(2)
get_net_buses(2)

gui_select_net_routing_of_selected

Select the shapes and vias of the nets related to the selected objects.

SYNTAX

status **gui_select_net_routing_of_selected**

DESCRIPTION

The **gui_select_net_routing_of_selected** command selects the routing for the nets of the currently selected objects. It is intended for interactive usage in the GUI for the application. Shapes and Vias that are already selected remain selected, while cell, pin, net, terminal, and port objects will have the routing for their nets selected.

EXAMPLES

Select the routing of the selected objects.

```
prompt> gui_select_net_routing_of_selected
```

SEE ALSO

change_selection(2)
get_shapes(2)
get_vias(2)

gui_select_net_shapes_of_selected

Select the shapes of the nets of the selected objects.

SYNTAX

status **gui_select_net_shapes_of_selected**

DESCRIPTION

The **gui_select_net_shapes_of_selected** command selects the shapes of the nets of the currently selected objects. It is intended for interactive usage in the GUI for the application. Shapes that are already selected remain selected, while cell, pin, net, terminal, and port objects will have the shapes of their nets selected.

EXAMPLES

Select the shapes of the nets of the selected objects.

```
prompt> gui_select_net_shapes_of_selected
```

SEE ALSO

change_selection(2)
get_shapes(2)

gui_select_net_vias_of_selected

Select the vias of the nets of the selected objects.

SYNTAX

status **gui_select_net_vias_of_selected**

DESCRIPTION

The **gui_select_net_vias_of_selected** command selects the vias of the nets of the currently selected objects. It is intended for interactive usage in the GUI for the application. Vias that are already selected remain selected, while cell, pin, net, terminal, and port objects will have the vias of their nets selected.

EXAMPLES

Select the vias of the nets of the selected objects.

```
prompt> gui_select_net_vias_of_selected
```

SEE ALSO

change_selection(2)
get_vias(2)

gui_select_nets_of_selected

Select the nets of the selected objects.

SYNTAX

status **gui_select_nets_of_selected**
[-hierarchical]

ARGUMENTS

-hierarchical

If this option is specified then all hierarchical nets of the original logical net are selected.

DESCRIPTION

The **gui_select_nets_of_selected** command selects the nets related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Nets that are already selected remain selected, while cell, pin, shape, via, terminal, port, and bundle objects will have their related nets selected.

EXAMPLES

Select the nets of the selected objects.

prompt> **gui_select_nets_of_selected**

SEE ALSO

change_selection(2)
get_nets(2)

gui_select_objects_of_selected_edit_group

Select the objects of the selected edit groups.

SYNTAX

status **gui_select_objects_of_selected_edit_group**

DESCRIPTION

The **gui_select_objects_of_selected_edit_group** command selects the objects of the currently selected edit groups. It is intended for interactive usage in the GUI for the application.

EXAMPLES

Select the objects of the selected edit group.

```
prompt> gui_select_objects_of_selected_edit_group
```

SEE ALSO

change_selection(2)
get_edit_groups(2)

gui_select_output_connections_of_selected

Select the output connections of the selected objects.

SYNTAX

status **gui_select_output_connections_of_selected**

DESCRIPTION

The **gui_select_output_connections_of_selected** command selects the output connections related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Ports and Pins that are already selected remain selected, while cell, net, shape, via, terminal, bound, pin_guide, pin_blockage, and design objects will have their related output connections selected.

EXAMPLES

Select the output connections of the selected objects.

```
prompt> gui_select_output_connections_of_selected
```

SEE ALSO

gui_select_input_connections_of_selected(2)
gui_select_connections_of_selected(2)
change_selection(2)
get_ports(2)
get_pins(2)

gui_select_port_buses_of_selected

Select the port buses of the selected objects.

SYNTAX

status **gui_select_port_buses_of_selected**

DESCRIPTION

The **gui_select_port_buses_of_selected** command selects the port buses related to the currently selected objects. It is intended for interactive usage in the GUI for the application. This command will select the port buses associated with the selected port.

EXAMPLES

Select the port buses of the selected objects.

```
prompt> gui_select_port_buses_of_selected
```

SEE ALSO

change_selection(2)
get_port_buses(2)

gui_select_ports_of_selected_power_supply_nets

Selects the ports of the selected power supply nets.

SYNTAX

status **gui_select_ports_of_selected_power_supply_nets**

DESCRIPTION

The **gui_select_ports_of_selected_power_supply_nets** command selects the ports and pins of blocks hierarchically related to the currently selected supply nets. The relationship is determined by **get_related_supply_nets** command. It is intended for interactive usage in the GUI for the application.

EXAMPLES

Select the ports, and pins of blocks selected supply nets.

```
prompt> gui_select_ports_of_selected_power_supply_nets
```

SEE ALSO

change_selection(2)
get_ports(2)
get_related_supply_nets(2)
get_supply_nets(2)

gui_select_power_domains_of_selected

Select the power domains of the selected objects.

SYNTAX

status **gui_select_power_domains_of_selected**

DESCRIPTION

The **gui_select_power_domains_of_selected** command selects the power domains related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Power domains that are already selected remain selected, while cells, voltage_areas, voltage area shapes will have their related power domains selected.

EXAMPLES

Select the power domains of the selected objects.

```
prompt> gui_select_power_domains_of_selected
```

SEE ALSO

change_selection(2)
get_power_domains(2)
get_voltage_areas(2)

gui_select_primary_power_supply_nets_of_selected

Select the primary power supply nets of the selected voltage areas, voltage area shapes, power domains.

SYNTAX

status **gui_select_primary_power_supply_nets_of_selected**

DESCRIPTION

The **gui_select_primary_power_supply_nets_of_selected** command selects the supply nets related to the currently selected voltage areas and power domains as primary power supply. It is intended for interactive usage in the GUI for the application. Voltage_areas, voltage area shapes, and power domains already selected will have their related primary power supply net selected.

EXAMPLES

Select the primary power supply nets of the selected objects.

```
prompt> gui_select_primary_power_supply_nets_of_selected
```

SEE ALSO

change_selection(2)
get_supply_nets(2)
get_voltage_areas(2)

gui_select_routing_corridors_of_selected

Select the routing_corridors of the selected objects.

SYNTAX

status **gui_select_routing_corridors_of_selected**

DESCRIPTION

The **gui_select_routing_corridors_of_selected** command selects the routing_corridors related to the currently selected objects. It is intended for interactive usage in the GUI for the application. If there are selected net, super net, pin, port, bundle, terminal, shape, or via objects the nets associated with those objects are used to determine the associated routing corridor.

EXAMPLES

Select the routing_corridors of the selected objects.

```
prompt> gui_select_routing_corridors_of_selected
```

SEE ALSO

change_selection(2)
get_routing_corridors(2)

gui_select_rp_blockages_of_selected

Select the relative placement (RP) blockages of the selected objects.

SYNTAX

status **gui_select_rp_blockages_of_selected**

DESCRIPTION

The **gui_select_rp_blockages_of_selected** command selects the relative placement (RP) blockages related to the currently selected objects. It is intended for interactive usage in the GUI for the application. When rp_group is selected, this command returns its RP blockage if any.

EXAMPLES

Select the RP blockages of the selected objects.

```
prompt> gui_select_rp_blockages_of_selected
```

SEE ALSO

change_selection(2)
get_rp_blockages(2)

gui_select_rp_groups_of_selected

Select the relative placement (RP) groups of the selected objects.

SYNTAX

status **gui_select_rp_groups_of_selected**

DESCRIPTION

The **gui_select_rp_groups_of_selected** command selects the relative placement (RP) groups related to the currently selected objects. It is intended for interactive usage in the GUI for the application.

When an RP blockage or cell is selected, this command returns the rp group that host the rp blockage or cell; when rp group is selected, this command returns the RP group inside it if any.

EXAMPLES

Select the RP groups of the selected objects.

```
prompt> gui_select_rp_groups_of_selected
```

SEE ALSO

change_selection(2)
get_rp_groups(2)
get_rp_group_objects(2)

gui_select_shapes_of_selected

Select the shapes of the selected objects.

SYNTAX

status **gui_select_shapes_of_selected**

DESCRIPTION

The **gui_select_shapes_of_selected** command selects the shapes of the currently selected objects. It is intended for interactive usage in the GUI for the application. Shapes that are already selected remain selected, while bound, voltage_area, routing_corridor, and port, objects will have their related shapes selected.

EXAMPLES

Select the shapes of the selected objects.

```
prompt> gui_select_shapes_of_selected
```

SEE ALSO

change_selection(2)
get_shapes(2)
get_bound_shapes(2)
get_routing_corridor_shapes(2)
get_voltage_area_shapes(2)

gui_select_shield_routing_of_selected

Select the shield routes and vias related to the selected signal nets, vias or routes.

SYNTAX

status **gui_select_shield_routing_of_selected**

DESCRIPTION

The **gui_select_shield_routing_of_selected** command selects the shield routing for the nets of the currently selected objects. It is intended for interactive usage in the GUI for the application.

EXAMPLES

Select the shield routing of the selected objects.

```
prompt> gui_select_shield_routing_of_selected
```

SEE ALSO

change_selection(2)
create_shields(2)
get_shapes(2)
get_vias(2)
gui_select_shielded_nets_of_selected(2)

gui_select_shielded_nets_of_selected

Select the signal nets that the selected objects are shielding.

SYNTAX

status **gui_select_shielded_nets_of_selected**

DESCRIPTION

The **gui_select_shielded_nets_of_selected** command selects the shielded nets related to the currently selected shield routes. It is intended for interactive usage in the GUI for the application.

EXAMPLES

Select the shielded nets of the selected shield routes.

```
prompt> gui_select_shielded_nets_of_selected
```

SEE ALSO

change_selection(2)
get_nets(2)
gui_select_shield_routing_of_selected(2)

gui_select_site_arrays_of_selected

Select the site arrays of the selected objects.

SYNTAX

status **gui_select_site_arrays_of_selected**

DESCRIPTION

The **gui_select_site_arrays_of_selected** command selects the site arrays related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Site arrays that are already selected remain selected, while site rows, tracks will have their related site arrays selected.

EXAMPLES

Select the site arrays of the selected objects.

```
prompt> gui_select_site_arrays_of_selected
```

SEE ALSO

change_selection(2)
get_site_arrays(2)
get_site_rows(2)
get_tracks(2)

gui_select_site_rows_of_selected

Select the site rows of the selected objects.

SYNTAX

status **gui_select_site_rows_of_selected**

DESCRIPTION

The **gui_select_site_rows_of_selected** command selects the site rows related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Site rows that are already selected remain selected, while site arrays will have their related site rows selected.

EXAMPLES

Select the site rows of the selected objects.

```
prompt> gui_select_site_rows_of_selected
```

SEE ALSO

change_selection(2)
get_site_rows(2)
get_site_arrays(2)

gui_select_supernets_of_selected

Select the supernets of the selected objects.

SYNTAX

status **gui_select_supernets_of_selected**

DESCRIPTION

The **gui_select_supernets_of_selected** command selects the supernets related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Supernets that are already selected remain selected, while nets, pins, ports, shapes, vias, bundles, and routing corridors will have their related supernets selected.

EXAMPLES

Select the supernets of the selected objects.

```
prompt> gui_select_supernets_of_selected
```

SEE ALSO

change_selection(2)
get_supernets(2)
get_pins(2)

gui_select_terminals_of_selected

Select the terminals of the selected objects.

SYNTAX

status **gui_select_terminals_of_selected**

DESCRIPTION

The **gui_select_terminals_of_selected** command selects the terminals related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Terminals that are already selected remain selected, while port, net, and design objects will have their related terminals selected.

EXAMPLES

Select the terminals of the selected objects.

```
prompt> gui_select_terminals_of_selected
```

SEE ALSO

change_selection(2)
get_terminals(2)

gui_select_topology_edges_of_selected

Selects the topology edges of the selected objects.

SYNTAX

status **gui_select_topology_edges_of_selected**

DESCRIPTION

The **gui_select_topology_edges_of_selected** command selects the edges related to the currently selected objects, with whom the same nets are shared. It is intended for interactive usage in the GUI for the application. Cell, net, terminal, port, pin, topology node, and topology edge objects will have their related topology edges selected.

EXAMPLES

Select the topology edges of the selected objects.

```
prompt> gui_select_topology_edges_of_selected
```

SEE ALSO

change_selection(2)
get_topology_edges(2)
get_topology_nodes(2)
gui_select_topology_nodes_of_selected(2)
gui_select_topology_plans_of_selected(2)
gui_select_topology_repeaters_of_selected(2)

gui_select_topology_nodes_of_selected

Selects the topology nodes of the selected objects.

SYNTAX

status **gui_select_topology_nodes_of_selected**

DESCRIPTION

The **gui_select_topology_nodes_of_selected** command selects the nodes related to the currently selected objects, with whom the same nets are shared. It is intended for interactive usage in the GUI for the application. Cell, net, terminal, port, pin, topology node, and topology edge objects will have their related topology nodes selected.

EXAMPLES

Select the topology nodes of the selected objects.

```
prompt> gui_select_topology_nodes_of_selected
```

SEE ALSO

change_selection(2)
get_topology_edges(2)
get_topology_nodes(2)
gui_select_topology_edges_of_selected(2)

gui_select_topology_plans_of_selected

Selects the topology plans of the selected objects.

SYNTAX

status **gui_select_topology_plans_of_selected**

DESCRIPTION

The **gui_select_topology_plans_of_selected** command selects the topology plans related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Topology node, edge, and repeater objects will have topology plans which own these objects selected.

If the objects are nets, supernets, or bundles, the command selects the topology plans that have one or more of those objects in their "objects" attribute collection.

EXAMPLES

Select the topology plans of the selected objects.

```
prompt> gui_select_topology_plans_of_selected
```

SEE ALSO

change_selection(2)
get_topology_edges(2)
get_topology_nodes(2)
get_topology_plans(2)
get_topology_repeaters(2)
gui_select_topology_edges_of_selected(2)
gui_select_topology_nodes_of_selected(2)
gui_select_topology_repeaters_of_selected(2)

gui_select_topology_repeaters_of_selected

Selects the topology repeaters of the selected objects.

SYNTAX

status **gui_select_topology_repeaters_of_selected**

DESCRIPTION

The **gui_select_topology_repeaters_of_selected** command selects the repeaters related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Topology plan and topology edge objects will have their related topology repeaters selected.

EXAMPLES

Select the topology repeaters of the selected objects.

```
prompt> gui_select_topology_repeaters_of_selected
```

SEE ALSO

change_selection(2)
get_topology_edges(2)
get_topology_nodes(2)
get_topology_repeaters(2)
gui_select_topology_edges_of_selected(2)
gui_select_topology_nodes_of_selected(2)
gui_select_topology_plans_of_selected(2)

gui_select_tracks_of_selected

Select the tracks of the selected objects.

SYNTAX

status **gui_select_tracks_of_selected**

DESCRIPTION

The **gui_select_tracks_of_selected** command selects the tracks related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Tracks that are already selected remain selected, while site arrays will have their related tracks selected.

EXAMPLES

Select the tracks of the selected objects.

```
prompt> gui_select_tracks_of_selected
```

SEE ALSO

change_selection(2)
get_site_arrays(2)
get_tracks(2)

gui_select_vmbucket

Select the Contents of a Visual Mode Bucket

SYNTAX

string **gui_select_vmbucket** -vmname *mode_identifier*
-name *bucket_identifier*
[-replace | -add | -remove]

string *mode_identifier*
string *bucket_identifier*

ARGUMENTS

-vmname *mode_identifier*

Specifies the visual mode to which the bucket is a member. The visual mode must already exist. To see the current list of defined visual modes, use the **gui_list_vm** command.

-name *bucket_identifier*

Specifies the name of the visual mode bucket.

-replace

Optional argument that indicates that the contents of the visual mode bucket will replace the selection list.

-add

Optional string that indicates that the contents of the visual mode bucket will be added to the selection list.

-remove

Optional string that indicates the contents of the visual mode bucket will be removed from the selection list.

DESCRIPTION

The **gui_select_vmbucket** command selects the contents of a visual mode bucket. The visual mode bucket is a member of a specific visual mode. The contents of the visual mode bucket can replace, be added to, or be removed from the selection list.

EXAMPLES

Select the 2nd bucket of SNAPSHOT visual mode.

```
shell> gui_select_vmbucket -vmname SNAPSHOT -name 1 -replace
```

SEE ALSO

gui_create_vm(2)
gui_get_vm(2)
gui_get_vmbucket(2)
gui_remove_vm(2)
gui_remove_vmbucket(2)
gui_set_vm(2)
gui_set_vmbucket(2)
gui_update_vm(2)

gui_select_voltage_areas_of_selected

Select the voltage areas of the selected objects.

SYNTAX

status **gui_select_voltage_areas_of_selected**

DESCRIPTION

The **gui_select_voltage_areas_of_selected** command selects the voltage_areas related to the currently selected objects. It is intended for interactive usage in the GUI for the application. Vias that are already selected remain selected, while voltage_area_shape objects will have their related voltage_areas selected.

EXAMPLES

Select the voltage_areas of the selected objects.

```
prompt> gui_select_voltage_areas_of_selected
```

SEE ALSO

change_selection(2)
get_voltage_areas(2)

gui_set_active_window

Make the specified window the active window

SYNTAX

status **gui_set_active_window**
-window *window_id*

Data Types

window_id string

ARGUMENTS

-window *window_id*

Specifies the window name id.

The matching window of this id will be made the active window.

Note: both toplevel windows and child windows of a particular toplevel window have a currently active window. If the window is a toplevel window then the currently active toplevel window is changed, of the window is a child window the currently active child window in the specified window's parent is changed.

DESCRIPTION

This command makes the specified window the active window.

EXAMPLES

The following example will make the toplevel window Toplevel.2 active.

shell> gui_set_active_window -window Toplevel.2

SEE ALSO

gui_close_window(2)
gui_exist_window(2)
gui_show_window(2)
gui_get_window_types(2)
gui_get_window_ids(2)
gui_create_window(2)
gui_get_current_window(2)

gui_set_attribute

Set the value of a GUI attribute for one or more objects.

SYNTAX

string **gui_set_attribute**
object_collection
attribute_name
attribute_value

string *object_collection*
string *attribute_name*
string *attribute_value*

ARGUMENTS

object_collection

A collection of one or more objects.

attribute_name

The name of a GUI attribute.

attribute_value

The value to be applied to the GUI attribute.

DESCRIPTION

Applies the given value to the GUI attribute for each element of the object collection. The return value is a list of one or more object names (depending on the size of the object collection) to which the attribute value has been applied. A Tcl error is raised if any error condition is encountered. This command is hidden.

EXAMPLES

psyn_shell-t> **gui_set_attribute [find cell U*7] some_attribute some_value**
/RISC_CORE/U7 /RISC_CORE/U17 /RISC_CORE/U27 /RISC_CORE/U37

SEE ALSO

gui_get_attribute(2)

gui_set_bucket_option

Sets the value for an option on a bucket in the specified visual mode or map mode.

SYNTAX

string **gui_set_bucket_option**
-map *map_name*
-bucket *bucket_name*
-option *option_name*
-value *value* | -default

Data Types

map_name string
bucket_name string
option_name string
value string

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

-bucket *bucket_name*

Specifies the name of the bucket.

-option *option_name*

Specifies the name of the option to be set.

-value *value*

Specifies the value for the option.

The **-value** option is mutually exclusive with the **-default** option. You must use one, but not both.

-default

Sets the option to its default value.

The **-default** option is mutually exclusive with the **-value** option. You must use one, but not both.

DESCRIPTION

This command sets the value for an option on a bucket in a visual mode or map mode. You must specify the visual mode or map mode name, the bucket name, and the option name. You must set the option to either a specific value or its default value. Use the **-value** option to set a specific value or use the **-default** option to set the option to its default value.

EXAMPLES

The following example sets the color of *bucket1* to yellow in the global route congestion map:

```
prompt> gui_set_bucket_option -map AREAPARTITION \
    -bucket bucket1 -option color -value yellow
```

SEE ALSO

gui_get_bucket_option(2)
gui_get_bucket_option_list(2)
gui_get_map_list(2)
gui_get_map_option(2)
gui_get_map_option_list(2)
gui_set_map_option(2)

gui_set_cell_block_marks

Sets a block mark on one or more cells.

SYNTAX

status **gui_set_cell_block_marks**
cells
block_mark

Data Types

cells list
block_mark string

ARGUMENTS

cells

Lists one or more cells to mark with a block mark, in a list of cell name patterns or cell collections.

block_mark

Specifies the block mark string value to be set for the listed cells.

DESCRIPTION

This command sets a block mark for one or more hierarchical or leaf-level cell instances. Typically, a short symbolic string name is used as the block mark. Ultimately, a block mark can appear in the GUI as the text label for a timing path category generated from a dynamic category rule. Such block marks should be kept short to avoid using up too much screen space in the GUI.

A common application involves marking certain interesting intellectual property (IP) blocks within a design. These block marks affect the calculation of the following timing path shell attributes, which are defined only when the GUI is running:

- **blocks** - An ordered block level path; includes start, through, and end blocks
- **through_blocks** - Includes through blocks, but not the start and end blocks
- **start_block** - The start block
- **end_block** - The end block
- **start_end_blocks** - A pair consisting of the start block and the end block, in that order

- **start_end_blocks_sorted** - A pair consisting of the start block and the end block, sorted alphabetically

These block marks also affect the calculation of the following cell shell attribute, which is defined only when the GUI is running:

- **block_mark** - String value of a cell instance's (explicit) block mark, or an empty string if there is no explicit block mark

In calculating the timing path shell attributes listed previously, a block is calculated for each timing point of a timing path. If the leaf cell instance on which a timing point resides is explicitly marked with a block mark, by using the **gui_set_cell_block_marks** command, the block for the timing point is the block mark for the leaf cell. If the leaf cell instance on which a timing point resides is not explicitly marked, the block for the timing point is the block mark on the first explicitly marked hierarchical cell located by traversing up the logical hierarchy from that leaf cell instance.

If no direct or indirect hierarchical parent cell is found to be explicitly marked, the block for the timing point has a default implicit string value of "_Top_". For a timing point that is a startpoint or endpoint of a timing path, where the startpoint or endpoint is an external design port, the block has a default implicit string value of "_Port_".

Note: Each timing point of a timing path always has a defined block, which can be an explicit block mark, "_Top_", or "_Port_".

The **through_blocks** attribute value can be an empty set of blocks, which is represented by the string value "_Empty_".

EXAMPLES

The following example sets the block mark to ALU for a hierarchical cell instance identified by a cell name:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU  ALU
```

The following example sets the block mark to ALU for a hierarchical cell instance identified by a nested run of the **get_cells** command:

```
prompt> gui_set_cell_block_marks [get_cells I_TOP/I_ALU]  ALU
```

The following example sets the block mark for a list of cell instances where the first item is identified by a cell name and the second item is identified by a nested run of the **get_cells** command:

```
prompt> gui_set_cell_block_marks [list I_ALU  [get_cells I_STACK_TOP/I1_STACK_MEM]] \
    my_mark
```

SEE ALSO

gui_get_cell_block_marks(2)
gui_remove_cell_block_marks(2)

gui_set_charts_data

Set charts data

SYNTAX

status **gui_set_charts_data**
{ -global | -model *model_id* | -view *view_name* |
-plot *plot_name* }
[-column *int*]
[{ -header | -row *int* }]
-name *string*
-value *string*

Data Types

model_id int
view_name string
plot_name string

ARGUMENTS

-global

Set global data.

-model *model_id*

Model to set data in.

-view *view_name*

View to set data in.

-plot *plot_name*

Plot to set data in.

-column *int*

Column to set model data in.

Only used by the *model* option.

-header

Set model header data.

Only used by the *model* option.

-row ***int***

Row to set model data in.

Only used by the *model* option.

-name ***string***

Name of data to set.

The supported names depend on which object the data is being set in:

For model the following names are supported:

value Set value of model horizontal header or row value.

column_type Set specific column type or all model column types. If the column is specified then this is the type for the specified column, if no column is specified this is a list of types for the model's columns. See **gui_create_charts_model** *column_type* for details of the syntax.

name Set model name.

For view the following names are supported:

fit fit plots in the view.

For plot no names are currently supported:

For global no names are currently supported:

-value ***string***

The value to set.

DESCRIPTION

Set data in model, view, plot or globally.

If model is specified then the *column* and *header* or *row* should be specified.

EXAMPLES

The following example sets the name of a model.

shell> gui_set_charts_data -model $model -name "Test Model"

SEE ALSO

gui_get_charts_data(2)

gui_set_charts_property

Set charts property

SYNTAX

status **gui_set_charts_property**
{ -view *view_name* | -plot *plot_name* |
-annotation *annotation_id* }
[-object *object_id*]
-name *string*
-value *string*

Data Types

view_name string
plot_name string
annotation_id string
object_id string

ARGUMENTS

-view *view_name*

View to set property in.

-plot *plot_name*

Plot to set property in.

-annotation *annotation_id*

Plot annotation to set property in.

-object *object_id*

Plot object to set property in.

The *plot* option must be specified as well.

-name *string*

Name of property to set.

The supported names depend on which object the data is being set in:

-value *string*

The value to set.

DESCRIPTION

Set property of view, plot or plot annotation.

EXAMPLES

The following example sets the line stroke color of a plot to red.

```
shell> gui_set_charts_property -plot $plot -name "stroke.color" -value red
```

SEE ALSO

gui_get_charts_property(2)

gui_set_current_category_tree

Sets the "current category tree" to the specified category tree

SYNTAX

gui_set_current_category_tree
category_tree

Data Types

category_tree name pattern or collection which resolves to exactly one category tree

ARGUMENTS

category_tree

A category tree name pattern or category tree collection which resolves to exactly one category tree.

DESCRIPTION

Sets the "current category tree" to the specified category tree. An empty string is returned.

The category tree name pattern (if specified) must be either the exact name of an existing tree, or (if there is only one existing category tree) the pattern "*", which means "all category trees" (or "the only category tree" in this case). More general forms of wild-card matching are not supported at this time.

At any time, exactly one of the existing category trees has the status "current category tree". When a new category tree is created, it immediately becomes the current category tree. If the current category tree is removed, a different category tree will be chosen as the current category tree. If there are no existing category trees, then there is no current category tree.

EXAMPLES

Here are examples of getting and setting the current category tree:

```
shell> get_attribute [gui_get_category_trees *] name
_tree008 _tree009 _tree010 _tree011
shell> get_attribute [gui_get_current_category_tree] name
_tree009
```

```
shell> gui_set_current_category_tree _tree008
shell> get_attribute [gui_get_current_category_tree] name
_tree008
```

SEE ALSO

gui_create_category_tree(2)
gui_remove_category_trees(2)
gui_get_category_trees(2)
gui_get_current_category_tree(2)
get_attribute(2)

gui_set_current_errors

Sets the current error data and the current error group in the Error Browser.

SYNTAX

string **gui_set_current_errors**
  [-data_name *ErrorDataName*]
  [-group *Group*]

ErrorDataName *String*
Group *String*

ARGUMENTS

-data_name *ErrorDataName*

ErrorDataName specifies the error data to make current. The error data name may be omitted. If omitted, makes all open error data current.

-group *Group*

Group specifies the group name, either a layer name or error type name, to make current. The group name may be omitted. If omitted, an entire error data file is made current.

DESCRIPTION

Sets the current error data and the current error group in the Error Browser. This is analogous to selecting error data and group in the Error Browser tree view. By default, there is are no current errors. The current errors determine the errors that are displayed in the error list and in the layout view by default.

If the error data option is omitted, selects the design, the parent of all open error data. The group option is valid only when there is a single error data specified, and raises an error if no error data was specified. The group string is interpreted as error type if error grouping has been set to **type**, else the group string is interpreted as layer. If the group option is omitted, all groups in the specified error data are made current.

EXAMPLES

The following example sets error grouping by error type, and sets the "Open Locator" type errors for LVS error data as the current

errors:

gui_set_error_browser_option -grouping type gui_set_current_errors -data_name LVS -group "Open Locator"

The following example sets the design as current, displaying all errors in all open error data in the error list and layout view.

gui_set_current_errors

SEE ALSO

gui_error_browser(2)
gui_set_selected_errors(2)

gui_set_current_task

Set current task to the given task.

SYNTAX

string **gui_set_current_task** -name *task_name*
-task *task_name*

string *task_name*

ARGUMENTS

-task *task_name*

Specifies the name of the task to make current.

DESCRIPTION

The gui_set_current_task command sets the current application task to the given task.

SEE ALSO

gui_create_task(2)
gui_remove_task(2)
gui_get_current_task(2)
gui_get_task_list(2)

gui_set_display_view

Sets the display view type (abstract, design, frame, layout, or outline) used in the layout when cell-specific views are active. Views are returned with the **gui_get_display_view** command.

SYNTAX

string **gui_set_display_view**
-cells *cell_list*
-view *view_name*

Data Types

cell_list list
view_name string

ARGUMENTS

-cells *cell_list*

Specifies the physical cells for which to set the view.

-view *view_name*

Specifies the view name. Accepted view names are abstract, design, frame, layout, outline, and the empty string {}.

DESCRIPTION

This command sets the display view type for the specified cell. The layout can display the contents of cells when the view level is set to a value greater than

1. Normally the displayed information is for the bound (or linked) view for the reference block. The layout also supports globally displaying an alternate view (for example the design view if the block is bound to abstracts) and a cell-specific view setting. Use this command to change the displayed view on a cell-by-cell basis.

If the specified view does not exist, then the layout will display the currently bound view.

This setting also controls the edit-in-place operation in the layout. The currently displayed view is always used.

EXAMPLES

The following example sets the design view for cell I_ORCA_TOP/I_BLENDER_4. When a block is bound to abstract views, you can display the design view for certain cells in order to fix DRC errors.

```
prompt> gui_set_display_view -cells I_ORCA_TOP/I_BLENDER_4 -view design
```

The following example resets the display view to the default by specifying an empty value.

```
prompt> gui_set_display_view -cells I_ORCA_TOP/I_BLENDER_4 -view {}
```

SEE ALSO

gui_get_display_view(2)

gui_set_error_browser_option

Sets options for the Error Browser dialog box.

SYNTAX

string **gui_set_error_browser_option**
[-grouping *Group*]
[-show_mode *ShowMode*]
[-view_mode *ViewMode*]
[-zoom_factor *ZoomFactor*]
[-hide_fixed *DoHide*]
[-hide_ignored *DoHide*]
[-advance_to_next_unfixed *DoNext*]
[-dim *DoDim*]
[-show_open_locator_nodes *DoShow*]
[-show_tooltip *DoShow*]
[-show_command_buttons *DoShow*]
[-highlight_objects *DoHighlight*]
[-auto_set_object_visible *SetVisible*]
[-auto_set_object_visible_type *SetVisibleType*]

Group *String*
ShowMode *String*
ViewMode *String*
ZoomFactor *float*
DoHide *Boolean*
DoNext *Boolean*
DoDim *Boolean*
DoShow *Boolean*
DoHighlight *Boolean*
SetVisible *Boolean*
SetVisibleType *String*

ARGUMENTS

-grouping *Group*

The **-grouping** option accepts argument values *type* or *layer* and sets whether errors are grouped by types or layers. This is similar to selecting or deselecting *Show by Layer* option in the Error Browser *Options* menu. By default, errors are grouped by type by default.

-show_mode *ShowMode*

The **-show_mode** option accepts argument values *all*, *selected*, or *none* and sets whether all errors, selected errors, are no

errors are shown in layout views. This is similar to making a selection from the *Show* field in the Error Browser. The default is that all errors are shown.

-view_mode *ViewMode*

The **-view_mode** option accepts argument values *zoom*, *pan*, or *off* and sets the layout view to zoom to, pan to, or not change to the currently selected errors in the Error Browser. This is similar to making a selection from the *Follow* field in the Error Browser. The default is to zoom to the selected error.

-zoom_factor *ZoomFactor*

The **-zoom_factor** option accepts float values between *0.1* and *10.0* and sets the zoom factor used when the view_mode is *zoom*. The zoom factor rounds to 1/10 precision. The default value is *1.0*.

-hide_fixed *DoHide*

The **-hide_fixed** option accepts Boolean values *1* or *true* to mean "hide", and *0* or *false* to mean "show". The option controls whether fixed errors are hidden or shown in the Error Browser and layout views. This is similar to selecting or deselecting *Hide Fixed Errors* option in the Error Browser *Options* menu. By default, fixed errors are shown.

-hide_ignored *DoHide*

The **-hide_ignored** option accepts Boolean values *1* or *true* to mean "hide", and *0* or *false* to mean "show". The option controls whether ignored errors are hidden or shown in the Error Browser and layout views. This is similar to selecting or deselecting *Hide Ignored Errors* option in the Error Browser *Options* menu. By default, ignored errors are shown.

-advance_to_next_unfixed *DoNext*

The **-advance_to_next_unfixed** option accepts Boolean values *1* or *true* to mean advance to next unfixed error when an error is fixed and *0* or *false* to advance to next fixed/unfixed error. This is similar to checking or unchecking the *Advance To Next Unfixed When Error is Fixed* option in the Error Browser. By default, unfixed errors are not advanced to.

-dim *DoDim*

The **-dim** option accepts Boolean values *1* or *true* to mean "dim" and *0* or *false* to mean "do not dim". The option governs whether layout views are dimmed when errors are displayed. This is similar to checking or unchecking the *Dim* option in the Error Browser. By default, layout views are not dimmed.

-show_open_locator_nodes *DoShow*

The **-show_open_locator_nodes** option accepts Boolean values *1* or *true* to mean "show", and *0* or *false* to mean "hide". This setting controls whether net nodes are highlighted together with the open locator flylines for selected open locator errors. This is similar to selecting or deselecting the *Display Net Nodes for Selected Open Locators* option in the Error Browser *Options* menu. By default, the tool does not show the net node highlights.

-show_tooltip *DoShow*

The **-show_tooltip** option accepts Boolean values *1* or *true* to mean "show", and *0* or *false* to mean "hide". This setting controls whether tool tips are shown on the error browser list view and details pane. By default, the tool shows the tool tips.

-show_command_buttons *DoShow*

The **-show_command_buttons** option accepts Boolean values *1* or *true* to mean "show", and *0* or *false* to mean "hide". This setting controls whether the command buttons at the bottom of the error browser are shown. By default, the tool shows the command buttons.

-highlight_objects *DoHighlight*

The **-highlight_objects** option accepts Boolean values *1* or *true* to mean "highlight", and *0* or *false* to mean "do not highlight". This setting controls whether design objects associated with the selected error are highlighted in the layout view using design object highlighting. When the error object selection changes, the highlight is automatically removed from the associated design

objects. By default, the tool does not highlights associated design objects.

-auto_set_object_visible *SetVisible*

The **-auto_set_object_visible** option accepts Boolean values *1* or *true* to mean "turn on" and *0* or *false* to mean "do not turn on". The option governs whether relevant object visibility of layout view are turned on when errors are displayed. This is similar to checking or unchecking the *Auto Set Object Visible* menu item in the 'Options' menu button of the Error Browser. By default, layout object visibility will be turned on for related error object display.

-auto_set_object_visible_type *SetVisibleType*

The **-auto_set_object_visible_type** option accepts argument values *incremental*, or *exclusive* to mean turn on layers/objects incrementally or exclusively when a different error is selected. This is similar to selecting the *Auto Set Object Visible Incremental/Exclusive* menu item in the 'Options' menu button of the Error Browser. By default, layout object visibility will be turned on incrementally for related error object display.

DESCRIPTION

This command sets option setting for the Error Browser. Multiple option values can be set using one command.

EXAMPLES

The following example sets the grouping of errors by layers, to show all errors, and to pan to selected errors.

prompt> **gui_set_error_browser_option -grouping layer -show_mode all -view_mode pan**

The following example sets the option to dim layout views when errors are displayed and to hide fixed errors:

prompt> **gui_set_error_browser_option -dim true -hide_fixed true**

SEE ALSO

gui_error_browser(2)
gui_get_error_browser_option(2)

gui_set_error_data_filter

Set a filter on open error data.

SYNTAX

string **gui_set_error_data_filter**
  -show | -hide
  [-types *TypeList*]
  [-layers *LayerList*]
  [-objects *ObjectCollection*]
  [-typed_object_names *ObjectSpecification*]
  [-object_types *ObjectTypeSpecification*]

TypeList *String List*
LayerList *String List*
ObjectSpec *String List or collection*
ObjectCollection *collection*
ObjectTypeList *String List*

ARGUMENTS

-show | -hide

Mutually exclusive required option to specify whether the filtering should hide the records matching the filter criteria (-hide) or prune to the records matching the filter criteria (-show).

-types *TypeList*

Optional argument to specify a list of type names as filter criteria. An error record would match this criteria if its type attribute value matches any type name in the list.

-layers *LayerList*

Optional argument to specify a list of layer strings as filter criteria. An error record would match this criteria if its layer string matches any layer string in the list. The layer strings is the layer_names string attribute value for the error. It is is also displayed in the error browser.

-objects *ObjectCollection*

Optional argument to specify a set of design objects as filter criteria. An error record can be associated with zero or more design objects,such as nets, pins, and cells. An error record would match this criteria if any of its associated objects matches any object in the collection. Associated objects for an error can be accessed using get_attribute. The attribute name is tool specific. For tools which restrict the types of object that may be associated with errors, the attribute name may be specific, such as "nets". For tools that support a more general object association, the attribute name may be more general, such as "objects".

-typed_object_names *ObjectSpecification*

Optional argument to specify a set of design objects as filter criteria. An error record can be associated with zero or more design objects, such as nets, pins, and cells. An error record would match this criteria if any of its associated objects matches any objects in the specification. Specify objects with type-name, object-name-list pairs, for example

{ net { NetName1 NetName2 NetName3 {} } }.

Multilple such pairs may be listed to specify objects of multiple types as filter criteria, for example

{ { net { Net1 Net2 Net3 {} } } { pin { PinA PinB } } }.

Associated objects for an error can be accessed using get_attribute. The attribute name is tool specific. For tools which restrict the types of object that may be associated with errors, the attribute name may be specific, such as "nets". For tools that support a more general object association, the attribute name may be more general, such as "objects". A null object association can be specified using an empty string as the object name.

-object_types *ObjectTypeList*

Optional argument to specify a list of object type names as filter criteria. An error record would match this criteria if the type attribute value of any of its associated objects matches any type name in the list.

Specify object types with type-name, type-name-list pairs, for example

{ net_type { Signal Power } }.

Multilple such pairs may be listed to specify lists for multiple object types as filter criteria, for example

{ { net_type { Signal Power } } { route_type { Detail User } } }.

DESCRIPTION

Set the given filter on open error data. Filter state is set and is applied on currently open error data and subsequently opened error data until cleared. When a new filter is set, the new filter overrides any previously set filter. There is no accumulation of filters.

In order to show errors that have been hidden by a previously set filter, clear the filter with gui_clear_error_data_filter.

You can either filter out errors matching the filter specification (use -hide) or prune the current errors to errors matching the filter specification (use -show).

If multiple kinds of criteria are used in a filter, then an error must satisfy each kind of criteria to be a match. For example, if a filter with both an error type list (the -types option) AND an object type list (the -object_types option) is set, then for an error to be a match for the filter, its type must be one of the types set in the filter AND its associated object type must be one of the object types set in the filter.

EXAMPLES

The following example illustrates how to show only errors of type "Short":

gui_set_error_data_filter -show -types "Short"

The following example illustrates how to show only errors of type "Short" that are also associated with the "Signal" net type:

gui_set_error_data_filter -show -types "Short" -object_types {{net_type "Signal"}}

The following example illustrates how to hide errors associated with the "Global" route type:

```
gui_set_error_data_filter -hide -object_types {{route_type {Global}}}
```

The following example shows how to clear the error data filter:

```
gui_clear_error_data_filter
```

SEE ALSO

gui_error_browser(2)
gui_clear_error_data_filter(2)
gui_set_error_browser_option(2)

gui_set_error_status

Set the error status value of one or more error records.

SYNTAX

string **gui_set_error_status**
-errors *Errors*
-status *Status*

Errors *String*
Status *Status*

ARGUMENTS

-errors *Errors*

Errors specifies the handle for a collection of errors on which to set the fixed state.

-status *Status*

The **-status** argument specifies the status value. The supported status values are tool dependent. All tools with physical error data for the error browser support values "error" and "fixed". Some tools also support "ignored".

DESCRIPTION

This command sets the status of the given error objects to the given status value.

The status of an error is displayed in the Error Browser and an error can be hidden or shown based on its status value. The status is a part of the textual report written to a file by gui_report_errors.

SEE ALSO

gui_error_browser(2)
gui_set_error_browser_option(2)
gui_report_errors(2)

gui_set_flat_hierarchy_color

Sets colors on all leaf cells descended from top-level virtual hierarchical cells or the specified virtual hierarchy in the current design.

SYNTAX

gui_set_flat_hierarchy_color
[*virtual_hierarchy_cell*]
-color *color_Id* | -cycle_color
[*-separator _separator*]
[*-threshold _threshold*]

Data Types

virtual_hierarchy_cell string
color_Id integer
_separator string
_threshold integer

ARGUMENTS

virtual_hierarchy_cell

Specifies a virtual hierarchial cell in the design.

-color *color_Id*

Specifies a Milkyway color ID, which is used to set the color on all leaf cells descended from the virtual hierarchical cell.

-cycle_color

Automatically generates a different color for each virtual hierarchical cell and sets the color on all leaf cells descended from virtual hierarchical cells.

-separator _separator

Specifies the delimiter used to determine the hierarchical structure of the flat netlist. The default delimiter is a slash (/).

-threshold _threshold

If the total number of leaf cells under a hierarchy is less than the specified threshold value then color will not be applied to hierarchy.

DESCRIPTION

The **gui_set_flat_hierarchy_color** command sets the colors on the leaf cells descended from top-level virtual hierarchical cells or the specified virtual hierarchy cells in the current design. Theses colors are used to display the virtual hierarchial cells in some graphic windows,such as layout view.

When both *virtual_hierarchy_cell* and *-color* are specified, the command sets the color_id on all leaf cells descended from the specified virtual hierarchical cell. If a threshold is also specified and the total number of leaf cells is less than the threshold, then color is not applied to any of the leaf cells descended from the heirarchy.

When both *virtual_hierarchy_cell* and *cycle_color* are specified, the command generates a random color and sets the color on all leaf cells descended from the virtual hierarchical cell. If a threshold is also specified and the total number of leaf cells is less than the threshold, then color is not applied to any of the leaf cells descended from the heirarchy.

When *-color* is specified without *virtual_hierarchy_cell* , the command sets the color on all the leaf cells in current design.

When *cycle_color* is specified without *virtual_hierarchy_cell*, the command generates a different color for each top-level virtual hierarchical cell in the current design and sets the colors on all leaf cells descended from the virtual hierarchical cells.

EXAMPLES

The following example sets color ID 8 on all the leaf cells under the virtual hierarchy H1/H2 in the current design,

```
prompt> gui_set_flat_hierarchy_color H1/H2 -color 8
```

The following example generates a different color for the virtual hierarchy H1/H2 and sets the color on all the leaf cells under the virtual hierarchy H1/H2 in the current design,

```
prompt> gui_set_flat_hierarchy_color H1/H2 -color_cycle
```

The following example sets color Id 8 on all leaf cells in the current design

```
prompt> gui_set_flat_hierarchy_color -color 8
```

The following example generates the command generates a different color for each top-level virtual hierarchical cell in the current design, and sets the color on all leaf cells descended from that hierarchical cell.

```
prompt> gui_set_flat_hierarchy_color -color_cycle
```

SEE ALSO

gui_unset_flat_hierarchy_color(2)

gui_set_hierview_data

Set various state in the current or named Hierarchy View and its sub-views.

SYNTAX

string **gui_set_hierview_data**
[-view *View_Name*]
[-tree_agroup *Attr_Name*]
[-tree_filter *String*]
[-map_area *Attr_Name*]
[-map_color *Attr_Name*]
[-childlist_name *String*]
[-childlist_agroup *Attr_Name*]
[-childlist_filter *String*]

View_Name String
Attr_Name String

ARGUMENTS

-view *View_Name*

Use the given HierView window name (ex: Hier.1) for the following actions. Default is to get the most recent used HierView window name.

-tree_agroup *Attr_Name*

Set the current Hier Tree attribute group name.

-tree_filter *String*

Set the current Hier Tree filter expression.

-map_area *Attr_Name*

Set the Hier Map Area attribute group name.

-map_color *Attr_Name*

Set the Hier Map Color attribute group name.

-childlist_name *String*

Set the current ChildList sub-view name.

-childlist_agroup *Attr_Name*

Set the current ChildList attribute group name.

-childlist_filter ***String***

Set the current ChildList filter expression.

DESCRIPTION

This command allows you to set various current Hierarchy View related view settings.

SEE ALSO

gui_get_hierview_data(2)

gui_set_highlight_options

Change the options that control highlighting.

SYNTAX

string **gui_set_highlight_options**

[-current_color *color_id* | -next_color | -auto_cycle_color *enable*]

string *color_id*
bool *enable*

ARGUMENTS

-current_color *color_id*

Changes the current highlight color to the specified value. The current highlight color is used as a default when no color is specified for some operations.

-next_color

Changes the current highlight color to the next color in the set of available colors. This will wrap around so that it can be used to cycle through the color set. The provided colors are yellow, orange, red, green, blue, purple, light_orange, light_red, light_green, light_blue, and light_purple.

-auto_cycle_color *enable*

Specify if the current color should automatically be incremented after each gui_change_highlight -add operation. The color is not advanced if an explicit color is specified with the -color option of gui_change_highlight.

DESCRIPTION

The gui_set_highlight_options command can be used to modify highlighting behavior.

EXAMPLES

Set the current highlight color to blue.

```
shell> gui_set_highlight_options -current_color blue
```

Enable auto cycling.

```
shell> gui_set_highlight_options -auto_cycle_color true
```

Advance to the next color.

```
shell> gui_set_highlight_options -next_color
```

SEE ALSO

gui_change_highlight(2)
gui_get_highlight(2)
gui_get_highlight_options(2)

gui_set_hotkey

Sets a key binding to a Tcl command or a menu command in a GUI window.

SYNTAX

string **gui_set_hotkey**
  -hot_key *key_name*
  -tcl_cmd *command_name* | -menu *menu_name*
  [-replace]
  [-delete]
  [-replay_log_only]
  [-window_type *window_type_name* | -all_window_types]

key_name string
command_name string
menu_name string
window_type_name string

ARGUMENTS

-hot_key *key_name*

Specifies the key that you are associating with the specified command. The *key_name* is a string that contains the key name can also contain one or more modifiers. The valid modifier keys are **SHIFT**, **ALT**, and **CTRL**. You specify the modifier names by prepending them to the key name and connecting them with a plus sign (+).

If you bind an unmodified key and the shift-modified key is not bound, the binding works for both the shifted and nonshifted keys. If you set separate bindings for the shift-modified key and the unmodified key, the bindings are unique. Note that shift modifiers work only with letter keys and function keys.

Menus display the first key binding defined for a menu item. The key binding always appears in uppercase with its modifiers. For example, an unmodified accelerator for the letter "a" is shown as "A' and a shift-modified "a" (an uppercase A) is shown as "SHIFT+A."

-tcl_cmd *command_name*

Specifies the tool command language (Tcl) code that the tool executes when you press the key and its modifiers, if any.

The **-tcl_cmd** option and the **-menu** option are mutually exclusive.

-menu *menu_name*

Specifies the menu command that the tool invokes if you press the key and its modifiers, if any, when the menu command is enabled.

The *menu_name* is a string that specifies the hierarchy of the menu labels, with the labels connected by arrows formed from a

hyphen and a greater than sign (->). For example, "File->Exit" specifies the Exit command on the File menu. The *menu_name* can also include the mnemonic for the menu text, which is specified with an embedded ampersand (&), but they are not required to do so.

The **-menu** option and the **-tcl_cmd** option are mutually exclusive.

-replace

Replaces an existing key binding for the specified menu command or Tcl command with the specified key.

-delete

Removes the key binding from the specified menu command or Tcl command.

-replay_log_only

Limits logging of the Tcl command associated with the key to the command replay log file. The tool suppresses the command echo and result display to the xterm and console, and the command does not appear in the output log or the Tcl history log.

The tool ignores this option if you do not specify the **-tcl_cmd** option.

-window_type *window_type_name*

Specifies the type of top-level window that the key should apply to. (A top-level window is a window with a menu bar.) If you do not specify this option, the tool uses the application default window type.

The read-only variable **gui_default_window_type(3)** specifies the application default window type.

-all_window_types

Sets the key binding in every type of window for which it is applicable.

For menu command key bindings, the tool searches all of the top-level windows for the specified menu command and adds the key binding in any window where it exists. If the tool does not find any applicable windows, it issues a warning.

For Tcl command key bindings, the tool adds the binding in every type of top-level window.

DESCRIPTION

This command binds a given key to a function in the graphical user interface (GUI). The function might be available on a menu or specified by Tcl code. A menu command can have multiple key bindings. Key bindings can be specified for built-in or user-defined menu commands.

Key bindings for menu commands have several additional features that Tcl command key binding do not have. The function for a menu command key binding is invoked only when the menu command is enabled. This ensures that the function is invoked only when it is valid to invoke the function. In addition, the primary key bindings for menu commands appear with the menu text to make it easier for you to remember the bindings.

EXAMPLES

The following example creates a menu item and adds an additional key binding to it.

```
prompt> gui_create_menu -menu "&Test->Sub&Menu->Echo Hi" \
```

```
  -tcl_cmd "echo hi" -hot_key "Ctrl+5"
prompt> gui_set_hotkey -menu "Test->SubMenu->Echo Hi" \
  -hot_key "Ctrl+Y"
```

The following example creates a key binding, P, that invokes a Tcl command to print a query on the currently selected objects.

```
prompt> gui_set_hotkey -tcl_cmd {query_objects [get_selection]} \
  -key P
```

SEE ALSO

gui_report_hotkeys(2)
gui_create_menu(2)
gui_delete_menu(2)
gui_create_toolbar(2)
gui_delete_toolbar(2)
gui_create_toolbar_item(2)
gui_delete_toolbar_item(2)
gui_default_window_type(3)
gui_custom_setup_files(3)

gui_set_layer_widths

Sets values in a list of widths per layer used for route editing.

SYNTAX

status **gui_set_layer_widths**
-layer *layer_name*
{ -clear | -values *layer_widths* }

Data Types

layer_name string
layer_widths list

ARGUMENTS

-layer *layer_name*

Specifies the routing layer for which to define the user widths.

-clear

Clears the list of user widths for the layer.

After the user widths are cleared, the *gui_get_layer_widths* command returns the default widths for the layer when the *-current* option is specified.

-values *layer_widths* }

Specifies a list of user widths to add to the current list of widths for the layer.

The list of layer widths is automatically sorted and uniquified.

If no user widths are currently defined, these values will be the only ones returned by *gui_get_layer_widths* when the *-current* option is specified.

DESCRIPTION

This command defines a list of discrete layer widths to be used for routing in the route editor.

EXAMPLES

The following example specifies the set of widths for the layer 'M1'.

prompt> **gui_set_layer_widths -layer M1 -values {0.1 0.2 0.4 0.8 1.6}**

SEE ALSO

gui_get_layer_widths(2)

gui_set_layout_layer_visibility

Set the visibility of layers in a layout window

SYNTAX

status **gui_set_layout_layer_visibility**
collection_of_layers
[-window *windowID*]
-toggle | -only

Data Types

string *windowID*

ARGUMENTS

collection_of_layers

The collection of layers or list of layer names provided in technology file.

-window *windowID*

This specifies the window to have its mode changed. If not specified then the currently active window will be used.

-toggle

Toggle the visibility of the specified layers. The option is mutually exclusive with -only.

-only

Turn off all layers and then turn on only the specified layers. The option is mutually exclusive with -toggle.

DESCRIPTION

This command is used to set visibility of layers in a layout view.

EXAMPLES

To turn on visibility of METAL2, METAL3 layers only:

```
shell-t> gui_set_layout_layer_visibility {METAL2 METAL3} -only -window Layout.1
1
```

To toggle visibility of layer METAL3:

```
shell-t> gui_set_layout_layer_visibility {METAL3} -toggle -window Layout.1
1
```

SEE ALSO

gui_set_setting(2)

gui_set_layout_user_command

Set user defined command for layout input.

SYNTAX

status **gui_set_layout_user_command**
-apply_cmd *TclCmd* | -clear
[-input_type *InputType*]
[-snap_type *SnapType*]
[-status_text *string*]
[-cancel_cmd *TclCmd*]

Data Types

TclCmd *string*
InputType *one of [rectangle | line | polygon | point]*
SnapType *one of [litho | site | midsite | wiretrack | midwiretrack | user]*

ARGUMENTS

-apply_cmd *TclCmd*

The user procedure to be called when input completed. The coordinates will be passed as an argument to this procedure. The options is mutually exclusive with -clear.

-clear

Cancel all pending input and return layout to default mouse mode. The options is mutually exclusive with -apply_cmd.

-input_type *InputType*

The desired input type: rectangle | line | polygon | point. Default is rectangle

-snap_type *SnapType*

The desired snap type: litho | site | midsite | wiretrack | midwiretrack | user Default is litho

-status_text *string*

The help string to be shown in layout window status bar.

-cancel_cmd *TclCmd*

The user procedure to be called when input is canceled.

DESCRIPTION

This command is used to facilitate creation of user defined tools to extend the layout view standard capabilities. The command sets layout view to given input mode and executes user callback procedure on input complete. The input points are passed as an argument to user callback.

EXAMPLES

Cut the object shape with user specified rectangle

```
proc my_cut_object_shape_proc { rect } {
  change_selection [cut_objects [get_selection] -bbox {$rect}]
}

gui_set_layout_user_command -apply_cmd {my_cut_object_shape_proc} -status_text "Drag the rectangle to cut selected objects"
1
```

SEE ALSO

change_selection(2)
get_selection(2)

gui_set_layout_visual_mode

Set the visual mode in a layout window

SYNTAX

string **gui_set_layout_visual_mode** [-mode *modeName*]
  [-window *windowID*]
  [-no_update]

string *modeName*
string *windowID*

ARGUMENTS

-mode *modeName*

This option specifies the name of the visual mode that should be shown. If not specified then it returns the view to its default coloring.

-window *windowID*

This specifies the window to have its mode changed. If not specified then the currently active window will be changed if it is a layout window, or it will chose a layout window if one exists.

-no_update

By default this command will also ask the visual mode to prompt for updating of its data. Specifying this option causes the command to switch to the new visual mode without updating its data.

DESCRIPTION

This command is used to change the current visual mode displayed in a layout view.

EXAMPLES

To set the layout view into snapshot visual mode:

shell-t> **gui_set_layout_visual_mode -mode snapshot**

Layout.1

To return the layout view to its default visual:

```
shell-t> gui_set_layout_visual_mode
Layout.1
```

SEE ALSO

gui_create_vm(2)

gui_set_map_option

Sets the value for an option in the specified visual mode or map mode.

SYNTAX

string **gui_set_map_option**
-map *map_name*
-option *option_name*
-value *value* | -default

Data Types

map_name string
option_name string
value string

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

-option *option_name*

Specifies the name of the option to be set.

-value *value*

Specifies the value for the option.

The **-value** option is mutually exclusive with the **-default** option. You must specify one, but not both.

-default

Sets the option to its default value.

The **-default** option is mutually exclusive with the **-value** option. You must specify one, but not both.

DESCRIPTION

This command sets the value for an option in a visual mode or map mode. You must specify the visual mode or map mode name and the option name. You must set the option to a specific value or its default value. Use the **-value** option to set a specific value or

the **-default** option to set the option to its default value.

EXAMPLES

The following example sets the number of bins to 9 in the global route congestion map:

```
prompt> gui_set_map_option -map AREAPARTITION \
    -option num_bins -value 9
```

SEE ALSO

gui_get_bucket_option(2)
gui_get_bucket_option_list(2)
gui_get_map_list(2)
gui_get_map_option(2)
gui_get_map_option_list(2)
gui_set_bucket_option(2)

gui_set_mouse_tool_option

Set an option on a mouse tool

SYNTAX

string **gui_set_mouse_tool_option** -tool *string*
-option *string*
-value *string*

string *string*
string *string*
string *string*

ARGUMENTS

-tool *string*

Tool to set option for.

-option *string*

Option to set.

-value *string*

New option value.

DESCRIPTION

This command sets a mouse tool option. Mouse tool options influence the operation of the mouse tool when that tool is active

EXAMPLES

> gui_set_mouse_tool_option -tool SELECT -option InputMode -value Rectangle

SEE ALSO

gui_mouse_tool(2)
gui_get_mouse_tool_option(2)

gui_set_performance_log_option

Configures performance log behavior of logging specific commands or all default commands

SYNTAX

status **gui_set_performance_log_option**
[-commands *command_names* | -log_all]
[-add | -remove]

Data Types

command_names list

ARGUMENTS

-commands *command_names*

This option is mutually exclusive -log_all. If -add or -remove are not given, the argument sets or refreshes the commands to be logged. If a logging process has not been started, this argument sets the specific commands to be logged for the upcoming logging process. If already been started, this argument refreshes the commands to be logged from now on.

If -add or -remove are given, the argument set the commands to be added to or removed from already existing commands list.

-add

This option is mutually exclusive with -remove and must be used with -commands. Use -add to add commands to the already existing commands list which is set by this command. If there is no existing commands list, this argument generates a commands list to be logged.

-remove

This option is mutually exclusive with -add and must be used with -commands. Use -remove to remove commands from the already existing commands list which is set by this command.

-log_all

This option is mutually exclusive with -commands. This option resets the commands to be logged as default.

DESCRIPTION

This command controls the performance log behavior of logging specific commands or all default commands. After execute option -commands, the default commands list is not effective until execute option -log_all. The commands to be logged can be reset as

default by option -log_all. This command can be used before and during a logging process and change the commands to be logged. After successfully execute this command, the logging behavior changes.

EXAMPLES

Configure log behavior which logs performance data before and after executing commands of gui_zoom and win_select_object.

prompt> **gui_set_performance_log_option -commands {gui_zoom win_select_objects}**

Add command gui_get_current_window to the already existing commands list.

prompt> **gui_set_performance_log_option -commands {gui_get_current_window} -add**

Remove command gui_get_current_window from the already existing commands list.

prompt> **gui_set_performance_log_option -commands {gui_get_current_window} -remove**

Reset the log behavior which logs performance data of all default commands.

prompt> **gui_set_performance_log_option -log_all**

SEE ALSO

gui_log_performance(2)
gui_get_performance_log_option(2)

gui_set_pref_value

Set the value of a preference key.

SYNTAX

string **gui_set_pref_value** -key *Key* -value *Value*
[-category *Category*]

Key *String*

Data Types

Category *String*

ARGUMENTS

-key *Key*

Key specifies the key to set the new value.

-value *Value*

Value specifies the new value of the key. The new value should be of the same data type, one of (integer,bool,double,color,string) as the data type of the value of the key.

-category *Category*

Category specifies the category that owns the specified key. If not specified, a default category will be used.

DESCRIPTION

This command set the value of a preference key. Returns the value of the preference key.

EXAMPLES

The following example create a user-defined category *some_category* for storing preferences and then create a boolean key *some_key* under that category with the initial value of false. The command **gui_set_pref_value** is then used to change the value of the preference key to true.

```
gui_create_pref_category -category some_category
gui_create_pref_key -category my_category -key some_key -value_type bool -value false
gui_set_pref_value -category my_category -key some_key -value true
```

SEE ALSO

gui_create_pref_category(2)
gui_create_pref_key(2)
gui_get_pref_value(2)
gui_get_pref_value_type(2)
gui_remove_pref_key(2)
gui_get_pref_categories(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_set_preset

Sets the specified preset for object specified by category

SYNTAX

status **gui_set_preset**
-category *category*
[-system]
[-shared]
-default
-name *name*

Data Types

category string
name string

ARGUMENTS

-category *category*

Specifies the name of the category that the presets be applied to.

-system

Apply given system preset. This option is mutually exclusive with the -shared, -default option.

-shared

Apply given shared preset. This option is mutually exclusive with the -system, -default option.

-default

Apply preference setting. This option is mutually exclusive with the -system, -shared, preset option.

-preset *preset*

Specifies the preset to be applied. This option is mutually exclusive with the -default option.

DESCRIPTION

This command applies the default or specified preset to a category.

EXAMPLES

The following example applies the system presets 'floorplan' to the Layout.

prompt> **gui_set_preset -category Layout -name floorplan -system**

SEE ALSO

gui_get_presets(2)

gui_set_region

Sets the coordinates of the current region rectangle or rectilinear polygon.

SYNTAX

status **gui_set_region**
shape

Data Types

shape list

ARGUMENTS

shape

Specifies the region coordinates.

For a rectangle, the values should be given in the following format:

{{x1 y1} {x2 y2}}

For a rectilinear polygon, the values should be given in the following format:

{{x1 y1} {x2 y2} ... {xN yN}}

DESCRIPTION

This command sets the region shape to be used in layout region mode and returns 1 if it is successful.

EXAMPLES

prompt> **gui_set_region {{-1743.945 18.357} {42.833 1523.657}}**
1

SEE ALSO

gui_get_region(2)

gui_set_select_menu_adds_to_selection

Controls the setting of the Select > Add to Selection menu item specifies whether Select > Related Object in the GUI replaces or adds to the current selection.

SYNTAX

gui_set_select_menu_adds_to_selection
-value true | false

ARGUMENTS

-value true | **false**

Specifies whether Select > Related Objects in the GUI replaces or adds to the current selection. Specify **-value true** to append to the current selection, **-value false** to replace the selection.

DESCRIPTION

This command controls the setting of the Select > Add to Selection menu item specifies whether Select > Related Object in the GUI replaces or adds to the current selection.

This command is logged when the Select > Add to Selection menu item is invoked.

EXAMPLES

The following example enables adding to the current selection.

prompt> **gui_set_select_menu_adds_to_selection true**

gui_set_selected_errors

Changes the selection in the Error Browser to add or replace errors.

SYNTAX

int **gui_set_selected_errors**
-add | -replace *errors*

Data Types

errors string

ARGUMENTS

-add *errors*

Adds the specified errors to the current selection. This option is mutually exclusive with the **-replace** option.

-replace *errors*

Replaces the specified errors in the current selection. This options is mutually exclusive with the **-add** option.

DESCRIPTION

Sets new selected errors or adds new selected errors in the Error Browser. Errors are identified as a collection of error objects. If the **-add** option is given, the command adds the given errors to the currently selected errors. If the **-replace** option is given, the command replaces the current selection with the given errors. The command fails if the given errors are not in the current set specified by the **gui_set_current_errors** command, or with a click in the Error Browser tree view. The command returns 1 if successful, 0 otherwise.

SEE ALSO

gui_clear_selected_errors(2)
gui_error_browser(2)
gui_set_current_errors(2)

gui_set_setting

Sets a setting on the specified window.

SYNTAX

gui_set_setting
-window *WindowID*
-setting *Setting*
-value *Value*

WindowID *String*
Setting *String*
Value *String* | *Bool* | *Int* | *Double*

ARGUMENTS

-window *WindowID*

WindowID specifies the window to set the setting

-setting *Setting*

Setting specifies the setting to set

-value *Value*

Value specifies the value of the setting

DESCRIPTION

Sets a setting on the specified window. Windows are views or top level windows. The setting is a property of the window and the value's type is specific to the given setting.

EXAMPLES

shell> **gui_set_setting -window Layout.1 -setting showCell -value true**

```
shell> gui_set_setting -window Layout.1 -setting viewLevel -value 1
```

SEE ALSO

gui_get_setting(2)

gui_set_task_list

Set the list of tasks that are visible in the task assistant, and set their order.

SYNTAX

gui_set_task_list
-tasks *TaskList*

TaskList *List*

DESCRIPTION

ARGUMENTS

-tasks *TaskList*

TaskList specifies the list of tasks that will be visible in the task assistant and their order. Tasks which are not included in the list will still exist but will not be shown in the selection combo box of the task assistant.

DESCRIPTION

The gui_set_task_list command sets the list of tasks that are visible and available in the task assistant and sets their order.

EXAMPLES

The following simple example sets the visible list of tasks to "Design Planning" followed by "Routing".

```
gui_set_task_list -tasks {{Design Planning} {Routing}}
```

SEE ALSO

gui_get_task_list(2)
gui_create_task(2)
gui_remove_task(2)

gui_set_timing_table_paths

Set timing paths into a specified timing table.

SYNTAX

status **gui_set_timing_table_paths**
[-command *ACmd*]
[-window *AWindow*]
[-new]
[-name *ATable*]

Data Types

ACmd string
AWindow string
ATable string

ARGUMENTS

-command *ACmd*

Specifies a tcl command that returns a collection of timing paths.

If no command is specified, the "Select Paths" dialog will appear, where the desired timing paths could be specified.

-window *AWindow*

Specifies a top level window id where the timing table is open.

If this option is not specified the most recently used top level window will be used.

-new

Creates a new timing table.

This option cannot be used with option -name at the same time.

-name *ATable*

Specifies an existing timing table by its window id.

This option cannot be used with option -new at the same time.

DESCRIPTION

The **gui_set_timing_table_paths** command sets a collection of timing paths into a specified timing table. The command allows to create new timing tables in a specified top level window, or setting a table with a collection of timing paths which are generated by a specified command.

EXAMPLES

The following example opens a new timing table in the most recently used top level window with fifteen timing paths.

```
shell> gui_set_timing_table_paths -new -command {get_timing_paths -max_paths 15}
```

This example shows how to update an existing timing table called TimingAnalysisDriver.3 with a command to show the selected timing paths.

```
shell> set my_paths [get_timing_paths -nworst 20 -delay_type max_rise]
shell> change_selection $my_paths
shell> gui_set_timing_table_paths -name TimingAnalysisDriver.3 -command { get_selection }
```

SEE ALSO

gui_get_window_ids(2)
get_timing_paths(2)

gui_set_utable_meta

Create a MetaData Object for use in creating a new UserTable.

SYNTAX

string **gui_set_utable_meta**
-name *MetaDataObj_Name*
[-remove]
[-title *Report_Title*]
[-comment *Report_Comment*]
[-tag *Report_Tag*]
[-menu *Root_Menu*]
[-col_name *Column_Name*]
[-col_type *Column_Type*]
[-read_only]
[-user_enums *Column_Type_Enums*]
[-user_list *Column_Type_List*]

MetaDataObj_Name String
Report_Title String
Report_Comment String
Report_Tag String
Root_Menu String
Column_Name String
Column_Type String
Column_Type_Enums List of Strings
Column_Type_List List of Strings

ARGUMENTS

-name *MetaDataObj_Name*

The name of the MetaData object.

-remove

Remove and free the given MetaData object.

-title *Report_Title*

Report title for a table.

-comment *Report_Comment*

Report comment for a table.

-tag *Report_Tag*

A tag string assigned to the table.

-menu *Root_Menu*

User Context Sensitive Menu Root name.

-col_name *Column_Name*

A column name that has the following column type.

-col_type *Column_Type*

A column type name.

-read_only

Mark the given column as read only which prevents editing data in that column.

-user_enums *Column_Type_Enums*

A list of single string values for a new user type defined by the -col_type argument.

-user_list *Column_Type_List*

A list of multi string values for a new user type defined by the -col_type argument. A table cell of this type can contain multiple unique values of this type separated by a comma.

DESCRIPTION

This command creates a new MetaData object that then can be used to add useful meta data to the creation of a new UserTable. A new UserTable is created via the gui_create_utable and the gui_import_utable commands. MetaData can be used to define certain column data/object types instead of using column name postfixes. Users can also define new data types that are made of a list of unique string values. A table cell can then be set to only allow single values (enums) of that new type or a comma separated list of values (lists) of that new data type. MetaData can also be exported via the gui_export_utable command along with the table data.

EXAMPLES

The following example creates a new MetaData object named myMetaObj which contains a report title and comment.

shell> **gui_set_utable_meta -name myMetaObj -title "My Report" -comment "This is a comment"**

The following example adds a definition of a new data type 'myData' assigned to the column 'myDataColumn'. This new data type has 4 values (valA valB valC and valD). Also optionally the user can add a help string after the value separated by a

colon ':' which is used to show tool tips.

```
shell> gui_set_utable_meta -name myMetaObj -col_name myDataColumn -col_type myData -user_enums {
  {valA: Help string for valA...}
  {valB: Help string for valB...}
  {valC: Help string for valC...}
  {valD: Help string for valD...}
}
```

The following example then imports a CSV file and adds the previously defined MetaData to that new table.

```
shell> gui_import_utable -file myTable.csv -meta_obj myMetaObj
```

SEE ALSO

gui_create_utable(2)
gui_export_utable(2)
gui_get_utable(2)
gui_import_utable(2)
gui_show_utable(2)

gui_set_var

Set the value of a typed tcl variable which was created with the gui_create_var command.

SYNTAX

string **gui_set_var** -name *VarName* -value *Value*

VarName *String*

Data Types

ARGUMENTS

-name *VarName*

VarName specifies the typed tcl variable to set the new value.

-value *Value*

Value specifies the new value of the variable. The new value should be of the same data type, one of (integer,bool,double,color,string,point,size,rect) as the data type of the value of the variable.

DESCRIPTION

This command set the value of a typed tcl variable that has a name and a data type, one of (integer,bool,double,color,string,point,size,rect). Typed tcl variables are created with the gui_create_var command. Returns the current value of the variable.

EXAMPLES

The following example create a typed tcl variable *my_var_1* and then set it to a new value using the command gui_set_var.

```
gui_create_var -name my_var_1 -value_type integer 320
gui_set_var -name my_var_1 -value 360
```

SEE ALSO

gui_create_var(2)
gui_get_var(2)
gui_remove_var(2)
gui_exist_var(2)

gui_set_vm

Sets visual mode attributes.

SYNTAX

string **gui_set_vm**
-name *identifier*
[-update_cmd *update_command*]
[-title *label*]
[-help_topic *subject*]
[-infotip *infotip*]
[-buckets *buckets*]
[-icon *string*]
[-netfilter *string*]
[-float *bool*]
[-set_exaggerations]
[-top_exaggeration *float*]
[-mid_exaggeration *float*]
[-bot_exaggeration *float*]
[-show_only_pins_of_nets *bool*]
[-enable_bucket_delete *bool*]

Data Types

identifier string
update_command string
label string
subject string
infotip string
buckets string
net_filter string
icon_spec string

ARGUMENTS

-name *identifier*

Specifies the name of the visual mode to be modified. This name is used as the argument when the associated visual mode commands requires a visual mode name to be specified.

-update_cmd *update_command*

Specifies an optional Tcl command that is executed when the visual mode is accessed. A typical use for this command is to update the state of the visual mode if its contents are likely to change under certain circumstances.

-title *label*

Specifies an optional title. The title can include embedded spaces. Use this option to provide a label for the visual mode in the GUI.

If you do not specify the **-title** option, the default title is an empty string and the tool uses the **-name** option value to provide the label.

-help_topic *subject*

Specifies the help topic text string. This text is used as the subdirectory to find a help page related to the visual mode.

-infotip *infotip*

Specifies the infoTip string.

-buckets *buckets*

Specifies the rendering order of the buckets. The tool renders the specified buckets from left to right based on their order in the list. Buckets that are not included in the list are rendered before any of the buckets that are specified in the list.

-icon *string*

Specifies the icon file name. Use this option to provide an icon for GUI menus.

-netfilter *string*

Specifies the net connection filtering.

-float true | false

Specifies whether the bucket contents have a floating point range. By default, the value is false.

-set_exaggerations

Specifies the bucket exaggerations.

-top_exaggeration *float*

Specifies the exaggeration for the top bucket, the value should be greater than or equal to zero. By default, the value is 0.

-mid_exaggeration *float*

Specifies the exaggeration for the middle bucket, the value should be greater than or equal to -1. By default, the value is -1.

-bot_exaggeration *float*

Specifies the exaggeration for the bottom bucket, the value should be greater than or equal to zero. By default, the value is 0.

-show_only_pins_of_nets true | false

Indicates that layout only show pins of net objects in buckets.

-enable_bucket_delete true | false

Specifies whether the buckets can be deleted from context menu. Only discrete visual mode buckets can be deleted. By default, the value is false.

DESCRIPTION

The **gui_set_vm** command modifies one or more attribute values of an existing visual mode.

EXAMPLES

The following example changes the title of a visual mode from "mycoloring1" to "test":

```
prompt> gui_set_vm -name mycoloring1 -title {test}
```

SEE ALSO

gui_create_vm(2)
gui_create_vmbucket(2)
gui_get_vm(2)
gui_get_vmbucket(2)
gui_remove_vm(2)
gui_remove_vmbucket(2)
gui_set_vmbucket(2)
gui_update_vm(2)

gui_set_vmbucket

Set attributes for Visual Mode Bucket

SYNTAX

string **gui_set_vmbucket** -vmname *mode_identifier*
-name *bucket_identifier*
[-infotip *infotip*]
[-netfilter *net_filter*]
[-color *color*]
[-pattern *pattern*]
[-exaggeration double]
[-number int]
[-maxval double]
[-minval double]
[-visible *visibility*]
[-title *label*]
[-collection *handle*]
[-above *bucket_identifier* | -below *bucket_identifier* | -at *top*|*bottom*]

string *mode_identifier*
string *bucket_identifier*
string *infotip*
string *net_filter*
string *color*
string *pattern*
string *visibility*
string *label*
string *handle*
string *bucket_identifier*
string *bucket_identifier*
string *top*|*bottom*

ARGUMENTS

-vmname *mode_identifier*

Specifies the visual mode to which the bucket is a member. The visual mode must already exist. To see the current list of defined visual modes use the gui_list_vm command.

-name *bucket_identifier*

Specifies the name of the visual mode bucket to be modified. To see the list of buckets associated with a specific visual mode use the gui_get_vm command with the -buckets option.

-infotip *infotip*

String for infotip.

-netfilter *net_filter*

Specifies net connection filtering.

-color *color*

Specifies bucket rendering color. Supports color naming by RGB triplet (for example: "#FFFF00" for yellow) as well as standard names (for example: "yellow").

-pattern *pattern*

Specifies bucket rendering fill pattern. Supported patterns are: SolidPattern, HorPattern, VerPattern, CrossPattern, BDiagPattern, FDiagPattern, DiagCrossPattern, Dense1Pattern, Dense2Pattern, Dense3Pattern, Dense4Pattern, Dense5Pattern, Dense6Pattern, Dense7Pattern, NoBrush.

-exaggeration double

Specifies bucket min pixel exaggeration value >= 0.

-number int

Specifies bucket display number value >= 0 that is used to provide a display number for the visual mode bucket in the gui. If the -number option is not specified, it will default to the number of objects in the bucket.

-maxval double

Specifies bucket maximum value.

-minval double

Specifies bucket minimum value.

-visible *visibility*

Specifies visibility state of bucket contents. Valid values are TRUE and FALSE.

-title *label*

Specifies an optional string (that can include embedded spaces) that is used to provide a label for the visual mode bucket in the gui. If the -title option is not specified, it will default to an empty string and the -name argument value value will be used to provide the labeling instead.

-collection *handle*

Tcl object collection to be colored by this visual mode bucket.

-above *bucket_identifier*

Specifies an optional visual mode bucket name above which the current bucket is to be rendered.

-below *bucket_identifier*

Specifies an optional visual mode bucket name below which the current bucket is to be rendered.

-at *top*|*bottom*

Specifies an optional string indicating whether the current visual mode bucket is to be rendered at the top or bottom of all other buckets. Valid values are top and bottom.

DESCRIPTION

The gui_set_vmbucket command modifies an instance of a visual mode bucket.

EXAMPLES

Change the color and pattern for the bucket named "cat1" in the visual mode "foo":

shell> **gui_set_vmbucket -vmname foo -name cat1 -color #00FF00 -pattern FDiagPattern**

Color the objects in the current selection red by placing them in the bucket named "cat1" in the visual mode "foo" and setting its bucket color:

shell> **gui_set_vmbucket -vmname foo -name cat1 -collection [get_selection] -color red**

SEE ALSO

gui_create_vm(2)
gui_create_vmbucket(2)
gui_get_vm(2)
gui_get_vmbucket(2)
gui_list_vm(2)
gui_remove_vm(2)
gui_remove_vmbucket(2)
gui_set_vm(2)
gui_update_vm(2)

gui_set_window_pref_key

Create a preference key owned by a particular window or window type

SYNTAX

new_value **gui_set_window_pref_key**
{ -window *window_id* | -window_type *window_type* }
[-category *category*]
-key *key*
-value_type *value_type*
-value *value*

Data Types

window_id string
window_type string
category string
key string
value_type string
value string

ARGUMENTS

-window *window_id*

Specifies the name of the window that the preference will be applied to. This option is mutually exclusive with the -window_type option.

-window_type *window_type*

Specifies the name of the window type that instances of the type will have the preference be applied to. This option is mutually exclusive with the -window option.

-category *category*

Specifies the category that the key belongs to. If not specified, a default category will be assigned.

-key *key*

Specifies the name of the key to create.

-value_type *value_type*

Specifies the the data type of the value of the key. It must be one of [bool | integer | double | color | string].

-value *value*

Specifies the value of the key. The value should be set appropriately in accordance with its value type. The bool type accepts [true, false, 0, 1, on, off]. The color type accepts a named color, eg. "red" or a string specifying the color in this format "#RRGGBB", for example, "#0000FF".

RETURNS

new_value

The new value of the key.

DESCRIPTION

This command creates a preference key with the specified key name or sets the value of an existing pref key. This key will have the specified value type and value, and it will be created under the specified category. If the category is not specified, the key will belong to a default category. Returns the value of the preference key.

EXAMPLES

The following example will create a window and set a preference on that window.

```
shell> set wnd [gui_create_window -type Console]
shell> gui_set_window_pref_key -window $wnd -category {caption} -key {title}
 -value_type string -value {Command Console}
```

SEE ALSO

gui_get_window_pref_value(2)
gui_exist_window_pref_key(2)
gui_remove_window_pref_key(2)
gui_get_window_pref_categories(2)
gui_get_window_pref_keys(2)
gui_get_window_pref_attribute(2)
gui_set_window_pref_attributes(2)
gui_create_pref_category(2)
gui_set_pref_value(2)
gui_get_pref_value(2)
gui_get_pref_value_type(2)
gui_remove_pref_key(2)
gui_get_pref_categories(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_update_pref_file(2)

gui_set_window_preset

Sets the specified preset for object specified by window_type

SYNTAX

status **gui_set_window_preset**
-window_type *window_type*
[-system]
[-shared]
-default
-preset *preset*

Data Types

window_type string
preset string

ARGUMENTS

-window_type *window_type*

Specifies the name of the window type that the presets be applied to.

-system

Apply given system preset. This option is mutually exclusive with the -shared, -default option.

-shared

Apply given shared preset. This option is mutually exclusive with the -system, -default option.

-default

Apply preference setting. This option is mutually exclusive with the -system, -shared, preset option.

-preset *preset*

Specifies the preset to be applied. This option is mutually exclusive with the -default option.

DESCRIPTION

This command applies the default or specified preset to a window type. This command has been deprecated, please use

gui_set_preset instead.

EXAMPLES

The following example applies the system presets 'floorplan' to the Layout.

prompt> **gui_set_window_preset -window_type Layout -preset floorplan -system**

SEE ALSO

gui_set_preset(2)
gui_get_presets(2)
gui_get_window_presets(2)

gui_show_charts_create_plot_dialog

Show dialog to interactively create plot from model

SYNTAX

status **gui_show_charts_create_plot_dialog**
{ -clct *collection* | -model *model_id* }
[-view *view_name*]

Data Types

collection string
model_id int
view_name string

ARGUMENTS

-clct *collection*

Collection to create plot from.

-model *model_id*

Model to create plot from.

-view *view_name*

View to add plot to. If not specified a new view will be created.

DESCRIPTION

Shows a dialog which allows the user to interactively create a plot from a model.

EXAMPLES

The following example shows the dialog for a model.

shell> gui_show_charts_create_plot_dialog -model $model

SEE ALSO

gui_show_charts_load_model_dialog(2)
gui_show_charts_view_model_dialog(2)

gui_show_charts_load_model_dialog

Show dialog to interactive load model from data file

SYNTAX

status **gui_show_charts_load_model_dialog**

Data Types

ARGUMENTS

DESCRIPTION

Shows a dialog which allows the user to interactively load a model from a data file.

EXAMPLES

The following example shows the dialog.

shell> gui_show_charts_load_model_dialog

SEE ALSO

gui_show_charts_create_plot_dialog(2)
gui_show_charts_view_model_dialog(2)

gui_show_charts_view_model_dialog

Show dialog with view of model or collection data

SYNTAX

status **gui_show_charts_view_model_dialog**
{ -clct *collection* | -model *model_id* }
[-columns *column_names*]

Data Types

collection string
model_id int
column_names string

ARGUMENTS

-clct *collection*

Collection to show.

-model *model_id*

Model to show.

-columns *column_names*

The names of the attributes to show columns for. If not specified all attributes are used.

DESCRIPTION

Shows a dialog which allows the user to view a model or collection.

EXAMPLES

The following example shows the dialog for a model.

shell> gui_show_charts_view_model_dialog -model $model

SEE ALSO

gui_show_charts_create_plot_dialog(2)
gui_show_charts_view_model_dialog(2)

gui_show_command_form

Generate and show a form dialog for a shell command.

SYNTAX

string **gui_show_command_form** -command *command*

string *command*

ARGUMENTS

-command *command*

The name of the shell command to show in the form.

DESCRIPTION

Create a form dialog for a shell command. The form contains fields to allow entry of option arguments.

The dialog provides the following operations for the command:

- execute the command
- append the command to a script
- show the man page for the command.

This command creates form dialogs for application defined commands. It will also create a form dialog for a public user defined tcl proc when its option value types have been defined using the **define_proc_attributes** command. The kinds of value input fields used in the form depends on the **type_name** attribute given for the option using the **-define_args** option. See man pages for **define_proc_attributes** and **gui_report_proc_arg_type_names** for more information.

This command is available only when the GUI is running.

EXAMPLES

The following example creates a form dialog for the command **gui_start**

```
shell> gui_show_command_form -command gui_start
1
```

The following is an example of a tcl proc definition followed by a **define_proc_attribute** command invocation to define its attributes. The proc is then shown in a form dialog with the **gui_show_command_form** command. The example proc accepts a net and routing layers as arguments.

```
shell> proc myProc {args} {
  parse_proc_arguments -args $args results
  foreach argname [array names results] {
    echo $argname = $results($argname)
  }
}
shell> define_proc_attributes myProc \
  -info "echo argument values" \
  -define_args { \
    {-net "select a net" "net" string \
         {required {type_name net}}} \
    {-layers "select routing layers" "routing layers" list \
         {optional {type_name {layer {subtype routing}}}}} \
  }
shell> gui_show_command_form -command myProc
```

SEE ALSO

define_proc_attributes(2)
gui_report_proc_arg_type_names(2)
gui_start(2)
gui_stop(2)

gui_show_file_in_editor

Show the contents of a file in an external text editor.

SYNTAX

gui_show_file_in_editor
-filename *filename*
[-line *linenum*]

Data Types

filename *string*
linenum *integer*

ARGUMENTS

-filename *filename*

filename specifies the name of the file to be edited.

-line *linenum*

linenum specifies the line in the file to start editing at.

DESCRIPTION

The gui_show_file_in_editor command opens a text editor to edit the specified file. The editor is a separate task so doen not block the current application.

By default the file is edited using the text editor from the EDITOR environment variable in a xterm. If the EDITOR environment variable is not set then the 'vi' text editor is used.

Both the editor and terminal (xterm) are searched for on the users path. If either the editor or terminal can no be found the command will fail.

EXAMPLES

The following example shows the file "test.txt" starting at line 100.

```
shell> gui_show_file_in_editor -file test.txt -line 100
```

SEE ALSO

gui_show_url_in_browser(2)

gui_show_man_page

Show a man page in the man browser

SYNTAX

string **gui_show_man_page** *topic [-apropos] [-html]*

string *topic*

ARGUMENTS

topic

Man page topic to be shown.

-apropos

Use the apropos command so search for commands that are related to the given topic.

DESCRIPTION

This command will launch or raise the man page browser, and display the topic specified in it. If no topic is specified, then the HOME page for man page indexes will be shown.

If the specified topic does not exactly match a given man page, then additional searching will be done for commands. If the topic matches one or more commands, then an index page will be generated showing all of the commands which match the specified topic pattern. If the topic doesn't match any commands, then we will use the topic as the initial part of a command and add a * to it and do command matching. Any completions for the command will be shown on an index page that can then be used to select a man page for viewing.

If the -apropos option is specified, then the topic will be searched via the apropos command and the topics returned will be shown with thier brief help strings, and provide links the the man pages referenced.

If the -html option is specified, the tool will display assoicated html file for given topic if the html file exists.

EXAMPLES

To show the man page for the find command:

gui_show_man_page find

To show the man page for all commands containing the word cell:
gui_show_man_page *cell*

To show the man page for the get_attribute command using command completion:
gui_show_man_page get_attr

SEE ALSO

man(2)
apropos(2)

gui_show_map

Displays or hides the specified visual mode or map mode.

SYNTAX

string **gui_show_map**
-map *map_name*
-show true | false
[-window *window*]

Data Types

map_name string
window string

ARGUMENTS

-map *map_name*

Specifies the name of the visual mode or map mode.

-show true | false

Displays the visual mode or map mode when set to **true**, or hides the visual mode or map mode when set to **false**.

-window *window*

Specifies the name of layout window in which to display or hide the visual mode or map mode.

DESCRIPTION

This command displays or hides the specified visual mode or map mode. If the **-window** option is not specified, the command uses the most recently active layout window.

EXAMPLES

The following example displays the highlight visual mode in the most recently active layout window:

```
prompt> gui_show_map -map Highlight \
    -show true
```

SEE ALSO

gui_get_map_list(2)

gui_show_palette

Shows the palette in the specified window state.

SYNTAX

status **gui_show_palette**
{ -name *palette_id* | -type *palette_type* }
[-parent *parent_name*]
[-show_state *window_state*]
[-dock_edge *dock_edge*]
[-size *{width height}*]

Data Types

palette_id string
palette_type string
parent_name string
window_state string
dock_edge string
width float
height float

ARGUMENTS

-name *palette_id*

Specifies the palette name id or title.

If the *-parent* option is specified only palettes in the specified parent toplevel window are shown. If the *-parent* option is not specified all palettes of this name in all toplevel windows are shown.

This option precludes the **-type** option

-type *palette_type*

Specifies the palette type id. Any matching palette of this type will be shown.

If the *-parent* option is specified only palettes in the specified parent toplevel window are shown. If the *-parent* option is not specified all palettes of this type in all toplevel windows are shown.

This option precludes the **-name** option

-parent *parent_name*

Specifies the parent toplevel window name.

-show_state *window_state*

Specifies the window state to show the palette.

Legal states are **maximized**, **minimized**, **normal**, or **docked**.

The default is **normal**.

-dock_edge *dock_edge*

Specifies the edge to dock the palette.

Legal dock edges are **left**, **top**, **right**, or **bottom**.

Note: this option is only valid if the state is set to **docked**.

The default is **right**.

-size *{width height}*

Specifies the width and height of the palette.

Note: this option is only valid if the state is set to **normal**.

DESCRIPTION

This command shows the specified palette in the specified state and optionally sets the geometry.

If no state is specified then it defaults to normal.

EXAMPLES

The following example shows the Layer palette docked on the left of the workspace.

```
shell> set top [gui_create_window -type TopLevel]
shell> set layout [gui_create_palette -type Layout -parent $top]
shell> gui_show_palette -name $layout -show_state docked -dock_edge left
```

SEE ALSO

gui_hide_palette(2)

gui_show_task_assistant

Show the task assistant.

SYNTAX

gui_show_task_assistant
-task *TaskName*
-item *ItemName*

TaskName *String*
ItemName *String*

ARGUMENTS

-task *TaskName*

TaskName specifies the name of the task to show in the task assistant.

-item *ItemName*

ItemName specifies the name of the task item to show in the task assistant.

DESCRIPTION

The gui_show_task_assistant command opens the task assistant window and displays the page for given task and task item.

SEE ALSO

gui_create_task(2)
gui_set_current_task(2)
gui_get_current_task_item(2)
gui_get_current_task_page(2)

gui_show_timing_paths

Shows the details of a collection of timing paths in a timing path inspector.

SYNTAX

gui_show_timing_paths
-paths paths
[-new]

ARGUMENTS

-paths

Collection of timing pathss to be inspected.

-new

Creates a new timing path inspector window.

DESCRIPTION

The command shows the details of a collection of timing paths in a timing path inspector. Unless requested by the user, this command will use the most recently used timing path inspector window if it exists. Otherwise it will create a new timing path inspector window.

EXAMPLES

The following example inspects a collection of timing paths.

prompt> **gui_show_timing_paths -paths $paths**

SEE ALSO

change_selection(2)

gui_show_toolbar

Displays the specified toolbar or all the toolbars in the specified window or for a window type.

SYNTAX

void **gui_show_toolbar**
-toolbar *tool_bar_name* | -all
[-window *window_id*]
[-window_type *window_type*]

Data Types

tool_bar_name string
window_id string
window_type string list

ARGUMENTS

-toolbar *tool_bar_name*

Specifies the toolbar you want to display. The **-toolbar** option and the **-all** option are mutually exclusive.

-all

Displays all the toolbars in the specified window. The **-all** option and the **-toolbar** option are mutually exclusive.

-window *window_id*

Specifies the window in which you want to display the specified toolbar or all toolbars. This option is mutually exclusive with the **-window_type** option. If both **-window** and **-window_type** options are omitted, then by default the tool displays the toolbar or toolbars in the active window.

-window_type *window_type*

Specifies the window types in which you want to display the specified toolbar or all toolbars. All existing windows of the given types will be affected. Toolbar visibility may be set on a window type before a window of the type exists. This option is mutually exclusive with the **-window** option. If both **-window** and **-window_type** options are omitted, then by default the tool displays the toolbar or toolbars in the active window.

DESCRIPTION

This command displays either all the toolbars or the toolbar with the specified name in the window with the specified window ID, or

in all windows of the given window type.

Toolbar visibility set by this command will override any visibility setting that has been saved and restored from previous sessions. The last toolbar visibility set by this command or by **gui_hide_toolbar** will be saved and restored on next invocation of the tool if save and restore has not been disabled.

EXAMPLES

The following example displays the File toolbar in the active window:

```
prompt> gui_show_toolbar -toolbar File
```

The following example displays all the toolbars in the layout window named LayoutWindow.1:

```
prompt> gui_show_toolbar -all -window LayoutWindow.1
```

The following example displays the File toolbar in all TimingWindow type windows:

```
prompt> gui_show_toolbar -all -window_type TimingWindow
```

The following example display all the toolbars in all window types:

```
prompt> gui_show_toolbar -all -window_type [gui_get_window_types -type toplevel]
```

SEE ALSO

gui_create_toolbar(2)
gui_delete_toolbar(2)
gui_create_toolbar_item(2)
gui_delete_toolbar_item(2)
gui_hide_toolbar(2)
gui_get_toolbar_names(2)
gui_get_window_types(2)

gui_show_url_in_browser

Show the contents of a URL in an external web browser.

SYNTAX

gui_show_url_in_browser
-url *URL*

URL *String*

ARGUMENTS

-url *Url*

Url specifies the url to be displayed.

DESCRIPTION

The gui_show_url_in_browser command an external web browser to display the specified web page. The default web browser is defined by the application but is usually 'firefox' on UNIX platforms.

If the web browser supports external comminication (which 'firefox' does) then if no web browser is running the web browser is started. If the web browser is running then the URL is loaded in the browser usually as a new tab.

The default browser can be set using the 'gui_online_browser' variable but the set of valid browsers is usually restricted by the application so not all browsers may be supported. If a unsupported browser is specified the command will fail.

SEE ALSO

gui_online_browser(3)
gui_show_file_in_editor(2)

gui_show_utable

Show the current loaded user tables in a UserTable view.

SYNTAX

string **gui_show_utable**
[-name *Table_Name*]
[-view *View_Name*]
[-filter *Filter_Expr*]

[-distribution] [-columns *Column_List*]
[-report] [-columns *Column*]

[-import *CSV_Filename* [-tsv_mode] [-ssv_mode] [-report_type]]
[-meta_obj *MetaObj_Name*]
[-open *utable_Filename* [-read_only]]

[-top_window]

Table_Name String
Filter_Expr String
Column String
Column_List StringList
View_Name String
CSV_Filename String
MetaObj_Name String
utable_Filename String

ARGUMENTS

-name *Table_Name*

The name of the user table to show inside the view. If not given, the view will show the default help page. The user can then select a table via the table selector located at the top of the view.

-view *View_Name*

The name of an existing UserTable view to activate.

-filter *Filter_Expr*

Apply the given filter expression to the opened table view. **Note:** This option is exclusive with the **-import** option. To use this command and filter option you must first use **gui_import_utable**.

-distribution

Show table in distribution mode for the column given by the '-columns' option.

-report

Show table in report mode for the list of columns given by the '-columns' option.

-columns *Column_List*

Provides a list of one or more column names for the previous 2 options.

-import *CSV_Filename*

The name of a CSV (comma seperated values) file to import and create a UserTable from. A column configuration form is presented to allow the user to adjust the final produced UserTable.

-tsv_mode

This flag changes the default delimiter from the comma char to the tab char.

-ssv_mode

This flag changes the default delimiter from the comma char to the semicolon char.

-report_type

This flag causes the reader to interpret the file as a report file and convert it into a user table. *Note:* Currently only the 'report_constraint -all_violators' report can be converted.

-meta_obj *MetaObj_Name*

Optional MetaData object name used to define table meta data. See the command gui_set_utable_meta for more information.

-open *utable_Filename*

The name of a UserTable native filename (.utable extension) to load into memory and present to the user. **Note**, this option is exclusive with the **-import** option.

-read_only

This flag can only be used with the **-open** option. It opens the file in a streaming read only manner, not copying all the data into memory which can save a lot of system memory if the file is huge.

-top_window

This flag will also open a new top level window to house the User Table view in.

DESCRIPTION

This command creates a new view (or reuse an existing view) to display and interact with your User Tables. For convenience, options are given for you to also import or open a new table from a file and display it.

User Tables provide a way for you to define and interact with your own tables of data. The data can persist on disk separate from the design database, and is completely under your control. User Tables can be generated from the attributes of a set of selected objects or they can come from external sources via an ASCII value file

in CSV/TSV/SSV format.

Interactively these tables can be viewed and support sorting, filtering, summary reporting, value distributions, editing, etc. as well as the ability to select/highlight objects in the design using the object names in columns identified as object type columns.

This command is available only when the GUI is running.

EXAMPLES

The following example just creates and opens a new UserTable view.

shell> **gui_show_utable**

The following example opens a new UserTable view and displays the contents of **my_table**.

shell> **gui_show_utable -name my_table**

The following example opens a new UserTable view and shows the import form used to configure the table being created from the values file **results.csv**. If no header line is supplied in the values file the configuration form will use default names that can be edited by the user before committing to table creation.

shell> **gui_show_utable -import results.csv**

The following example opens a new UserTable view and connects to the UserTable file **my_table.utable** in a read only streaming mode for display.

shell> **gui_show_utable -open my_table.utable -read_only**

SEE ALSO

gui_append_utable(2)
gui_change_selection_utable(2)
gui_close_utable(2)
gui_create_utable(2)
gui_export_utable(2)
gui_get_utable(2)
gui_import_utable(2)
gui_set_utable_meta(2)
gui_open_utable(2)
gui_write_utable(2)

gui_show_window

Show the window in the specified window state

SYNTAX

status **gui_show_window**
-window *window_id*
[-show_state *window_state*]
[{ -rect *rect* | -size *isize* }]

Data Types

window_id string
window_state string
rect {{x1 y1} {x2 y2}}
isize {width height}

ARGUMENTS

-window *window_id*

Specifies the toplevel window or view id to show.

-show_state *show_window_state*

Specifies the window state to show the toplevel window or view.

For a toplevel window we have one of:

normal - show at preferred size
maximized - show maximized (fill whole screen)
minimized - show iconized
hidden - hide but do not destroy

Note: The window manager may or may not fully honor the specified state. See your window manager documentation for more details.

For a view window we have one of:

normal - show at preferred size in view area and raise to top
maximized - show maximized in view area and raise to top
minimized - show iconized in view area
hidden - hide but do not destroy

Note: If displaying maximized then any existing view windows are also maximized. If displaying minimized or normal then any maximized windows are shown as normal.

The default is **SHOW_WINDOW_STATE_NORMAL**.

-rect ***rect***

Specifies the size and position of the window.

It is of this format "{{x1 y1} {x2 y2}}", where {x1 y1} specifies the top left location and {x2 y2} specifies the bottom right location of the window.

Note: This option precludes the use of the *-size* option.

-size ***isize***

Specifies the width and height of the window.

It is of this format "{width height}"

Note: This option precludes the use of the *-rect* option.

DESCRIPTION

This command shows the specified toplevel window or view in the specified state and optionally sets the geometry.

If no state is specified then it defaults to normal.

EXAMPLES

The following examples will create a layout window then use this command to illustrate the various options

```
shell> set top [gui_create_window -type TopLevel]
shell> set layout [gui_create_window -type Layout -parent $top]
shell> gui_show_window -window $layout -show_state {maximized}
shell> gui_show_window -window $layout -show_state {normal}
```

SEE ALSO

gui_close_window(2)
gui_exist_window(2)
gui_create_window(2)
gui_get_window_types(2)
gui_get_window_ids(2)
gui_set_active_window(2)
gui_get_current_window(2)

gui_sort_charts_model

Sort charts model

SYNTAX

status **gui_sort_charts_model**
-model *model_id*
-column *column_name*
[-decreasing]

Data Types

model_id int
column_name string

ARGUMENTS

-model *model_id*

Model to sort.

-column *column_name*

Column name or number to sort.

-decreasing

Whether to sort descreasing instead of the default increasing.

DESCRIPTION

Sort column in charts model.

EXAMPLES

The following example sorts the second column of a model in decreasing order.

shell> gui_sort_charts_model -model $model -column 1 -decreasing

SEE ALSO

gui_create_charts_model(2)

gui_start

Starts the application GUI.

SYNTAX

string **gui_start**
[-file *script*]
[-no_windows]
[-offscreen 1|0]
[-- x_args ...]

string *script*

ARGUMENTS

-file "*script*"

The given script file is sourced before the GUI starts.

-no_windows

The GUI starts without showing the default window.

-offscreen

If the specified value is 1 then the GUI starts in offscreen mode.

If the specified value is 0 then the GUI starts in normal (X Display) mode.

-- x_args ...

X11-specific arguments to pass to the X connection.

DESCRIPTION

This command starts the application GUI from the shell prompt. It is ignored if the application GUI has already been started. start_gui is an alias for gui_start.

Note the application can only ever connect to one X server during program execution, so changing the value of the DISPLAY environment variable after the first successful gui_start command has no effect.

EXAMPLES

The following example starts the application GUI.

shell> **gui_start**

SEE ALSO

gui_stop(2)
start_gui(2)
stop_gui(2)

gui_stop

Stops the application GUI.

SYNTAX

string **gui_stop**

ARGUMENTS

None.

DESCRIPTION

This command stops the application GUI and returns to the shell prompt. It is ignored if the application GUI has not been started or has been stopped. stop_gui is an alias for gui_stop.

EXAMPLES

The following example stops the application GUI and returns to the shell prompt.

shell> **gui_stop**

SEE ALSO

gui_start(2)
start_gui(2)
stop_gui(2)

gui_trim_dangling_wires

Trims dangling wire ends of selected wires or nets.

SYNTAX

status **gui_trim_dangling_wires**
[*object_list*]

Data Types

object_list collection

ARGUMENTS

object_list

Specifies a collection or selection set containing objects to trim. If this option is not specified, the global selection set is used.

DESCRIPTION

This command trims dangling ends of selected net shapes, net shapes of selected nets, selected route corridor shapes, or route corridor shapes of selected route corridors.

Only net shapes and route corridor shapes with connections at both ends will be trimmed. Unconnected shapes or shapes only connected at a single point will be ignored. You can manually delete these shapes if required.

Trimming of dangling net shapes on power and ground nets is limited to selected net shapes. Net shapes of selected power and ground nets will be ignored. This option allows you to do some minor repairs but reduces the chance that you will accidently try to repair the whole power mesh which could be slow and is better achieved using separate power planning commands.

This command honors the 'is_fixed' attribute. Any fixed net shapes or route corridor shapes are ignored unless 'Ignore fixed attribute' is set in the Edit Options dialog.

EXAMPLES

The following example trims dangling ends for all net shapes on the net N1.

prompt> **change_selection [get_nets N1]**

```
prompt> gui_trim_dangling_wires
```

This example uses an alternative syntax with a collection.

```
prompt> gui_trim_dangling_wires [get_nets N1]
```

SEE ALSO

change_selection(2)
get_edit_setting(2)
get_selection(2)
set_edit_setting(2)

gui_unset_flat_hierarchy_color

Removes colors set on leaf cells.

SYNTAX

gui_unset_flat_hierarchy_color
[*virtual_hierarchy_cell*]
[*-separator _separator*]

Data Types

virtual_hierarchy_cell string
_separator string

ARGUMENTS

virtual_hierarchy_cell

Specifies a virtual hierarchial cell in the design.

-separator _separator

Specifies the delimiter used in determining hierarchical structure of the flat netlist. The default delimiter option is a slash (/).

DESCRIPTION

The **gui_unset_flat_hierarchy_color** command removes the colors set on leaf cells. The uncolor cells are used to display the cells on some graphic windows, such as layout view.

If *virtual_hierarchy_cell* is specified, the command removes the colors set on all leaf cells descended from the virtual hierarchical cell in the current design and restores their original colors.

If *virtual_hierarchy_cell* is not specified, the command removes colors on all leaf cells in the current design.

EXAMPLES

The following example removes colors set on all the leaf cells in the current design,

prompt> **gui_unset_flat_hierarchy_color**

The following example removes colors set on all the leaf cells under the virtual hierarchy H1/H2 in the current design,

prompt> **gui_unset_flat_hierarchy_color H1/H2**

SEE ALSO

gui_set_flat_hierarchy_color(2)

gui_update_attrgroup

Updates a group of attributes for an object type.

SYNTAX

status **gui_update_attrgroup**
-class *design_object*
-name *name*
[-attr_list {{*attr_1*}...{*attr_n*}}]
[-add]
[-delete]
[-move up | down | top | bottom | after | before]
[-attr *attribute*]
[-anchor *attribute*]

Data Types

design_object string
name string
attr_1 string
attr_n string
attribute string

ARGUMENTS

-class *design_object*

Specifies the type of design object.

-name *name*

Specifies the non-editable name of the attribute group to update for the specified object type.

-attr_list {{*attr_1*}...{*attr_n*}}

Lists and orders the attributes to be included in the group.

-add

Adds the attribute specified by the **-attr** option to the end of attributes list, by default, or after the attribute specified by the **-anchor** option.

-delete

Removes the attributes specified by the **-attr** option from the attribute list. Note that the attribute still exists.

-move up | **down** | **top** | **bottom** | **after** | **before**

Moves the attribute specified by the **-attr** option in the specified direction. If you specify **before** or **after** to move the attribute relative to the position of a reference attribute, use the **-anchor** option to specify the reference attribute.

-attr ***attribute***

Specifies the attribute to be added, removed, or moved with the **-add**, **-delete**, or **-move** option.

-anchor ***attribute***

Specifies the reference attribute to be used with the **-add** or **-move** option.

DESCRIPTION

The **gui_update_attrgroup** command updates an attribute group for the specified object type. You can reset the entire attribute list by using the **-attr_list** option, or you can change the list by adding, removing, or moving with a single attribute.

The order of the attributes in an attribute group is important and is set by the **-attr_list** option when you create or update the group.

EXAMPLES

```
prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \
  -attr_list { "TestAttr1" "TestAttr2" "TestAttr3" }

prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \
  -add -attr "TestAttr5"

prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \
  -add -attr "TestAttr5" -anchor "TestAttr2"

prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \
  -delete -attr "TestAttr5"

prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \
  -move down -attr "TestAttr2"
```

SEE ALSO

gui_create_attrgroup(2)
gui_delete_attrgroup(2)
gui_list_attrgroups(2)

gui_update_pref_file

Save current application preferences to the user preference file.

SYNTAX

string **gui_update_pref_file** [-file *FilePathName*]

FilePathName *String*

ARGUMENTS

-file *FilePathName*

FilePathName specifies the full pathname of the user preference file to write the current preferences to. If this option is missing, the file to write to is controlled by the gui variables : pref_file_name and pref_file_path. These two variables are set with the gui_set_var command and their values retrieved with the gui_get_var command.

DESCRIPTION

This command updates the user preference file. Normally, application will automatically save user preferences to a default system preference file on exit, and retrieve the preferences from the default user preference file on application start. So, this command is rarely used, except internally.

SEE ALSO

gui_create_pref_category(2)
gui_create_pref_key(2)
gui_set_pref_value(2)
gui_get_pref_value(2)
gui_get_pref_value_type(2)
gui_remove_pref_key(2)
gui_get_pref_categories(2)
gui_get_pref_keys(2)
gui_exist_pref_category(2)
gui_exist_pref_key(2)
gui_set_var(2)
gui_get_var(2)

gui_update_vm

Update Visual Mode

SYNTAX

string **gui_update_vm** -name *identifier*

string *identifier*

ARGUMENTS

-name *identifier*

Specifies the visual mode name whose associated tcl update command will be executed. The tcl update command is associated with the visual mode using the -update_cmd option to the gui_create_vm or gui_set_vm commands.

DESCRIPTION

Execute a visual mode's associated tcl update command.

EXAMPLES

To force the visual mode "vm1" to execute its associated update command, use the following:

shell> **gui_update_vm -name vm1**

SEE ALSO

gui_create_vm(2)
gui_create_vmbucket(2)
gui_get_vm(2)
gui_get_vmbucket(2)
gui_list_vm(2)
gui_remove_vm(2)

gui_remove_vmbucket(2)
gui_set_vmbucket(2)
gui_set_vm(2)

gui_update_vm_annotations

Updates visual mode annotations.

SYNTAX

string **gui_update_vm_annotations**
-clear
-center
-add *points*
-draw_net *style*
[-type *shape*]
[-text *message*]
[-color *color*]
[-pattern *pattern*]
[-line_style *line_style*]
[-width *width*]
[-info_tip *info_tip*]
[-query_text *query_text*]
[-query_command *query_command*]
object

Data Types

points list
style string
shape string
message string
color string
pattern string
line_style string
width integer
info_tip string
query_text string
query_command string
object string

ARGUMENTS

-clear

Removes all annotations from the specified object.

-center

Interprets all object locations as the centers of the objects rather than their origins.

-add *points*

Adds points for the annotation shape that you specify with the **-type** option. The points are specified by a tool command language (Tcl) list in which each element is one of the following:

- the {x y} location of a point
- a collection containing a single object that has a visual representation in the layout. In other words, it is not a net, for example. The location of the point is the origin of the object.

If a collection is specified the list element can optionally include a position type, one of:

- center : the annotation is placed at the center of the largest rectangle of the object. The largest rectangle is the rectangle with the most area which is completely contained inside the object shape.
- bbox_center : the annotation is placed at the center of the bounding box of the object.
- bbox_ll : the annotation is placed at the lower left of the bounding box of the object.
- bbox_lr : the annotation is placed at the lower right of the bounding box of the object.
- bbox_ul : the annotation is placed at the upper left of the

  bounding box of the object.
- bbox_ur : the annotation is placed at the upper right of the bounding box of the object.

If a bounding box position type is specified, i.e. one of 'bbox_center', 'bbox_ll', 'bbox_lr', 'bbox_ul' or 'bbox_ur', then the list element can also optionally include an x and y fractional offset from this bounding box point. The x and y offset is a fraction of the width and height of the object's bounding box respectively. For example, for a 'bbox_center' position type, an x offset of -0.5 is the left edge and an x offset of 0.5 is the right edge. Similarly a y offset of -0.5 is the bottom edge and a y offset of 0.5 is the top edge.

For annotations referring to a single-object collection, if the object is moved, the tool updates the point automatically as needed.

-draw_net *style*

Specifies how the nets are displayed in the layout view for this annotation if the specified objects are nets or physical buses.

This annotation overrides the current net display setting on the View Settings panel. The valid *style* values are

flyline
routing
routing_flyline

-type *shape*

Specifies the type of annotation. The valid *shape* values are

rect
line
arrow
polygon
polyline
text

-text *message*

Specifies a text string that you want to display.

-color *color*

Specifies the color. The default is the bucket color.

-pattern *pattern*

Specifies the fill pattern. The default is none.

-line_style *line_style>*

Specifies the line style. The default is none.

-width *width*

Specifies the line width. The default is 1.

-info_tip *info_tip*

Specifies an info tip for this annotation. When the user starts the query tool in the layout, and puts the mouse cursor over this annotation the specified text will be displayed as the infoTip.

If the text starts with a '=' character the text after it will be considered to be a tcl command which, when executed, will return the query text to be displayed. The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

%window the layout window name
%view the layout view name
%object a collection containing the annotation

-query_text *query_text*

This option is only valid if an info tip was specified. Use this to specify a more detailed string that is displayed in the query palette when the object is selected via the query tool. If query text is not specified then the query tool will just use the info tip string.

If the text starts with a '=' character the text after it will be considered to be a tcl command which, when executed, will return the query text to be displayed. The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

%window the layout window name
%view the layout view name
%object a collection containing the annotation

-query_command *query_command*

This option is only valid if an info tip was specified. Use this to specify a tcl command to be run when the user clicks on the annotation.

The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

%window the layout window name
%view the layout view name
%object a collection containing the annotation
%x the x position clicked in design coordinates
%y the y position clicked in design coordinates

object

Specifies the object with which to associate the annotation.

DESCRIPTION

This command adds annotations to objects returned by the **gui_create_vm_objects** command.

Layout view visual modes are used to color objects in the layout view based on certain criteria. You can use the **gui_update_vm_annotations** command to display not only object coloring but some additional annotations for the objects in a visual mode bucket.

EXAMPLES

To create a bucket that displays cells and a line connecting those cells to the location (0,0), enter code like the following example:

```
prompt> set objs [gui_create_vm_objects $cells]
  foreach_in_collection o $objs {
    set c [get_attribute $o core_obj]
    gui_update_vm_annotations $o -type line -add [list {0 0} $c]
  }
prompt> gui_create_vmbucket -vmname $vm -name $bucket -color red -collection $objs
```

SEE ALSO

gui_create_vm(2)
gui_create_vmbucket(2)
gui_create_vm_objects(2)

gui_view_port_history

Does the gui_view_port_history operations.

SYNTAX

gui_view_port_history
[-window *window_name*]
[-next]
[-previous]
[-add *name*]
[-to *name*]
[-list_names]
[-rect *bounding box*]
[-delete_name]
[-isprevious]
[-isnext]
[-dialog]
[-tcl_list]
[-help_cmd]

ARGUMENTS

-window *fwindow_name*

Give the name of a window. this is required option except if you use *-dialog* option.

-next

Specify if you want to go to previous stored view port history. If you use *-next* option you can not use any other option except -*window* option: they are mutually exclusive.

-previous

Specify if you want to go to previous stored view port history. If you use *-previous* option you can not use any other option except -*window* option: they are mutually exclusive.

-add *name*

Current view port will be stored by this name. *-add* option can be combined with *-rect* option. In that case the view port history with the given rect instead of current view port will be saved in to view port history with the given name.

If you use *-add* option you can not use any other option except *-window* and *-rect* option : they are mutually exclusive.

-to *name*

Give the name of a view port history already stored. If you use *-to* option you can not use any other option except -*window* option:

they are mutually exclusive.

-list_names

Specify if you want to see list of names of all the named view port history. If you use *-list_names* option you can not use any other option except *-window* option: they are mutually exclusive.

-delete_name *name*

The already stored view port history with this name will be deleted. If you use *-delete_name* option you can not use any other option except *-window* option: they are mutually exclusive.

-rect *boundingbox*

Adds give the bounding box (in design units) to unnamed view port history. The values should be given in following format. format: {{x1 y1} {x2 y2}} *-rect* option can be combined with *-add* option. In that case the view port history with the given rect instead of current view port will be saved in to view port history with the given name. If you use *-add* option you can not use any other option except *-window* and *-add* option : they are mutually exclusive.

-isnext

return true returns true if there previous zoom history. If you use *-isnext* option you can not use any other option except *-window* option: they are mutually exclusive.

-isprevious

returns true if there next zoom history. If you use *-isprevious* option you can not use any other option except *-window* option: they are mutually exclusive.

-dialog

specify to invoke dialog for named view port history. If you use *-dialog* option you can not use any other option except *-help_cmd* option: they are mutually exclusive.

-tcl_list

return a tcl_list of named view port history If you use *-tcl_list* option you can not use any other option except *-window* option: they are mutually exclusive.

help_cmd *help_command*

Give a command which will be executed when help buttonis clicked in the dialog.

DESCRIPTION

This command store view port for the application. At least one option needs to be specified.

last 100 View ports will be stored in view port history, which can be accessed by specifying **-previous** and **-next** options.

When you use **-add** then the current view port will be stored by the given name in the to view port history.

when you use **-to** then the view port stored by this name will be shown and added to the view port history.

EXAMPLES

The following example zooms in to the given rectangle.

fpc_shell> **gui_view_port_history -rect {{-2232.321 -536.887} {141.286 2175.807}}**

The following example zooms to next .

fpc_shell> **gui_view_port_history -next**

The following example zooms to previous.

fpc_shell> **gui_view_port_history -previous**

The following example gives list of all the named zooms.

fpc_shell> **gui_view_port_history -list_names**

The following example puts the current zoom in to zoom history by given name.

fpc_shell> **gui_view_port_history -add xyz**

The following example zooms to already stored to zoom .

fpc_shell> **gui_view_port_history -to xyz**

The following example deletes an already stored to zoom.

fpc_shell> **gui_view_port_history -delete xyz**

gui_write_annotations

Saves annotations to a Tcl file.

SYNTAX

status **gui_write_annotations**
-file *filename*
[-append]
[-force]
[-compress *method*]
[*annotations*]

Data Types

filename string
method string
annotations collection

ARGUMENTS

-file *filename*

Specifies the name of the file in which the tool saves the annotation script.

-append

Append the annotation script to an existing file instead of creating a new one. The specified file must exist. This option cannot work with *-force*.

-force

Overwrite the specified file, if it exists, with the new one. This option cannot work with *-append*.

-compress *method*

Specifies the compression type to use when writing the file. By default, no compression is performed.

annotations

Specifies a collection of annotations to write. Get a collection of annotations with the gui_get_annotations command. If this option is not specified, then all the annotations with **global** window handle and **global** group type would be written by, annotations with specific window handle or group type are excluded.

DESCRIPTION

This command writes a collection of annotations to a Tcl script.

EXAMPLES

Write out all global annotations.

prompt> **gui_write_annotations -file my_annotations.tcl**

Write out all annotations in Layout.1.

prompt> **gui_write_annotations -file my_annotations.tcl** \
[gui_get_annotations -window Layout.1]

SEE ALSO

gui_add_annotation(2)
gui_get_annotations(2)
gui_remove_all_annotations(2)
gui_remove_annotations(2)

gui_write_category_script

Generates and writes out a categorization script.

SYNTAX

gui_write_category_script
-file *file_name*
[-overwrite]
[-tree *tree_id*]

Data Types

file_name sting
tree_id string

ARGUMENTS

-file *file_name*

File name of the output file to which to write the generated categorization script.

-overwrite

Allows an existing file to be overwritten. If you do not specify this option, an existing file cannot be overwritten, and an error message is generated.

-tree *tree_id*

String ID of the category tree in which the write category script operation occurs; the current category tree is used if the *-tree* option is omitted.

DESCRIPTION

This command generates and writes out a categorization Tcl script that captures the state of the specified category tree. The generated script includes needed commands, if any, to recreate any user-defined non-built-in category rules that the categorization scheme depends on. The generated script is suitable for replaying in a subsequent run of the tool by sourcing the script using the **source** command.

Before replaying the script it is recommended that a similar design and set of conditions (as existed when the categorization scheme was first created) are set up first. The command returns "1" if successful; otherwise, an empty string is returned.

You should run this command only if at least one category tree is available. The command acts on the category tree specified.

EXAMPLES

The following example shows how to generate a category replay script and write it to the category_script.tcl file:

```
shell> gui_write_category_script -file category_script.tcl
1
```

EXAMPLES

The following example shows the categorization script replaying what was previously saved using the **gui_write_category_script** command. The most recently used category tree is recreated.

```
shell> source category_script.tcl
1
```

SEE ALSO

gui_create_category(2)
gui_remove_category(2)

gui_write_charts_data

Write charts data

SYNTAX

status **gui_write_charts_data**
-view *view_name*
[-file *file_name*]

Data Types

view_name string
file_name string

ARGUMENTS

-view *view_name*

View to output data for.

-file *file_name*

Filename to output data to. If not specified then the the output is sent to the terminal.

DESCRIPTION

Write charts data as a tcl script to terminal or a file.

This command allows the user to save the state of the plots in the current view to a tcl command which can be source'd to reproduce the plot.

Details of the models loaded for the view's plots, the views annotations, the plots and the plots annotations are saved.

Only the view and plot properties that have changed from their default values are saved.

Note: Only the filename, tcl variable or collection name of the model data can be saved so these external dependencies, file data, contents of the tcl variable, or contents of the collection will need to be recreated before the script can be used.

EXAMPLES

The following example writes view data to the file "view_data.tcl".

```
shell> gui_write_charts_data -view Charts.1 -file "view_data.tcl"
```

SEE ALSO

gui_create_charts_model(2)

gui_write_charts_image

Save charts plot or view to image file

SYNTAX

status **gui_write_charts_image**
-view *view_name*
-plot *plot_name*
-file *string*

Data Types

view_name string
plot_name string

ARGUMENTS

-view *view_name*

Charts view to save (all charts).

-plot *plot_name*

Single charts plot in view to save.

-file *string*

Output filename.

The file suffix is used to determine type i.e. .png for PNG file or .svg for SVG file.

If the file suffix (converted to lower case) is not .png or .svg then PNG is used.

DESCRIPTION

Save charts plot or view to PNG or SVG image file.

EXAMPLES

The following example saves all charts in a view to a PNG file charts.png.

```
shell> gui_write_charts_image -view $view -file charts.png
```

SEE ALSO

gui_create_chart(2)

gui_write_hierarchy_colors

Writes current hierarchy Module Boundary status and colors to a file.

SYNTAX

status **gui_write_hierarchy_colors**
-file *file_name*

Data Types

file_name string

ARGUMENTS

-file *file_name*

Specifies the name of file to write.

DESCRIPTION

The **gui_write_hierarchy_colors** command writes current hierarchy Module Boundary status and colors to a file. Next time, you can source this file to restore hierarchy Module Boundary status and color.

EXAMPLES

The following examples writes current Module Boundary status and colors to a data file named color.tcl.

prompt> **gui_write_hierarchy_colors -file color.tcl**

SEE ALSO

gui_explore_logic_hierarchy(2)

gui_write_timing_paths

Writes a persistent path data file containing a timing path collection.

SYNTAX

status **gui_write_timing_paths**
-file *file_name*
[-overwrite]
[-tag *tag_name*]
[-comment *comment*]
paths

Data Types

file_name string
tag_name string
comment string
paths collection

ARGUMENTS

-file *file_name*

Specifies the name of the persistent path data file to write. By default, if the file already exists, the command exits without writing the file.

-overwrite

Specifies that file that already exists will be overwritten.

-tag *tag_name*

Specifies a tag name for the timing path.

-comment *comment*

Specifies a user comment for the timing path file.

paths

Specifies the collection of timing paths to write.

DESCRIPTION

The **gui_write_timing_paths** command writes a persistent path data file containing the given path collection. The persistent path data file contains all the information needed to re-create those timing paths using the **gui_read_timing_paths** command.

EXAMPLES

The following examples writes out a timing path collection to a persistent path data file named my.paths.

```
prompt> gui_write_timing_paths $paths -file "my.paths"
```

SEE ALSO

gui_read_timing_paths(2)

gui_write_utable

Write the UserTable to file in native UserTable format.

SYNTAX

string **gui_write_utable**
-name *Table_Name*
[-file *File_Name*]
[-filter *Filter*]
[-read_only]
[-overwrite]

Table_Name String
File_Name String

ARGUMENTS

-name *Table_Name*

The name of the user table to write out to a file.

-file *File_Name*

The name of the file to write the table to, by default the name of the table is used with the '.utable' extension.

-filter *Filter*

This argument allows you to filter the rows in the given table to only rows that match the filter. The filter expression is the same expression allowed in the GUI filter field of the UserTable GUI view.

-read_only

This flag closes the table that has been written out and then re-opens it in read-only (streaming) mode which can save a lot of system memory if the table is huge.

-overwrite

This flag will force the destination file to be overwritten.

DESCRIPTION

This command writes the given table name to a file in the native user table format, preserving all meta data (filter, sort order and column information). The UserTable file can then be later re-opened to copy the information back into memory or opened in a read-only (streaming) mode using the command **gui_open_utable**.

EXAMPLES

The following example writes out the given table to a file name equal to the table name plus the extension '.utable'.

```
shell> gui_write_utable -name my_table
```

The following example writes out the given table (overwrites the file if necessary) and then closes the table in memory and re-opens it in read-only (streaming) mode.

```
shell> gui_write_utable -name my_table -read_only -overwrite
```

SEE ALSO

gui_append_utable(2)
gui_change_selection_utable(2)
gui_close_utable(2)
gui_create_utable(2)
gui_export_utable(2)
gui_import_utable(2)
gui_open_utable(2)
gui_show_utable(2)

gui_write_window_image

Saves an image of a view window or toplevel window in the specified image file

SYNTAX

status **gui_write_window_image**
-file *filename*
[-format *image_format*]
[-window *window_name*]
[-size *window_size*]

Data Types

filename string
image_format string
window_name string
window_size {width height}

ARGUMENTS

-file *filename*

Specifies the name of the file in which the tool saves the window image.

If you include a file name extension, it determines the image format unless you also specify the *-format* option.

-format *png* | *xpm* | *jpg* | *bmp*

Specifies the image format to be used in the file. The default value is png.

Note that if the format that you specify with the *-format* option is different from the format specified by the file name extension, the *-format* value takes precedence and the tool appends an additional extension to the file name.

-window *window_name*

Specifies the name of the window for which you want to save an image in the specified image file.

You can view a list of the window names for all the open toplevel and view windows by using the *gui_get_window_ids* command.

By default, the tool saves an image of the most recently active view window.

-size *window_size*

The size of the image to be generated. If not specified then the full window size is used.

DESCRIPTION

The **gui_write_window_image** takes a snapshot of a specified toplevel window or child view window in a specified image format and writes the result to a specified file.

EXAMPLES

The following example writes a snapshot of the TopLevel.1 window to a image file "snapshot.png".

```
shell> gui_write_window_image -window TopLevel.1 -file snapshot.png
```

The following example writes a snapshot of the Layout.1 view to a image file "layout.png" of size 800 pixels by 600 pixels.

```
shell> gui_write_window_image -window Layout.1 -file layout.png -size {800 600}
```

SEE ALSO

gui_get_window_ids(2)
gui_get_current_window(2)

gui_zoom

Change the viewport of a view

SYNTAX

gui_zoom
-window *window*
[-fit | -exact]
[-full]
[-selection]
[-rect *{{lx ly} {ux uy}}*]
[-factor *factor*]
[-at_point *{x y}*]
[-clct *clcf*]
[-zoom_in_mode]
[-zoom_out_mode]
[-select_mode]

window *String*
rect *String*
factor *Float*

ARGUMENTS

-window *window*

window specifies the window of which this command should change the viewport.

-full

Zoom such that all objects are visible.

-factor *factor*

Keep the center of the viewport and zoom by factor *factor*. A *factor*" > 1 causes a zoom-in, a *factor*" < 1 causes a zoom-out. The argument must be greater than zero.

-at_point *{x y}*

Can only be specified with -factor argument. If supplied, zoom is performed at specified point instead of viewport center. The option is not supported by all window types.

-fit

Effects the -rect, -selection and -clct arguments. If supplied, will cause the zoom to not over zoom. Will zoom a reasonable

distance away from the rectangle or objects being zoomed. The definition of "reasonable" is window dependent. This argument is mutually exclusive with -exact.

-exact

Effects the -rect, -selection and -clct arguments. If supplied, will cause the zoom to zoom exactly to the arguments. This argument is mutually exclusive with -fit.

-selection

Zoom such that all selected objects that are represented in the window are visible and if possible are easy to see and are shown in a reasonable context. If neither -fit or -exact is supplied, -fit is assumed.

-rect *{{lx ly} {ux uy}}*

Zoom such that the window shows the rectangle *rect* in its viewport. What coordinates are assumed for *rect* are dependent on the window. If neither -fit or -exact is supplied, -exact is assumed.

-clct *clct*

Zoom such that all objects in *clct* that are represented in the window are visible and if possible, are easy to see and are shown in a reasonable context. If neither -fit or -exact is supplied, -fit is assumed.

-zoom_in_mode

Put the window in zoom-in mode.

-zoom_out_mode

Put the window in zoom-out mode.

-select_mode

Put the window in selection mode.

DESCRIPTION

The **gui_zoom** controls the viewport of a view that supports zooming. Not all views need to support all of the functionality provided by the interface of this command.

EXAMPLES

The following example has the layout view show the entire design and then zooms in by a factor of 2, i.e. only a quarter of the area of the design is still visible:

```
gui_zoom -window Layout.1 -full
gui_zoom -window Layout.1 -factor 2
gui_zoom -window Layout.1 -rect {{0.0 0.0} {123.456 500.123}}
gui_zoom -window Layout.1 -rect {{0.0 0.0} {123.456 500.123}} -fit
gui_zoom -window Layout.1 -selection -exact
```

SEE ALSO

gui_zoom_all_layouts_to_current_view

Rescales all layout views to the current view.

SYNTAX

gui_zoom_all_layouts_to_current_view

ARGUMENTS

None.

DESCRIPTION

The command sets the zoom factor of all layout views to the zoom factor of current layout view.

EXAMPLES

prompt> **gui_zoom_all_layouts_to_current_view**

SEE ALSO

gui_zoom(2)

gui_zoom_to_selected_errors

Layout window zoom to errors selected in the error browser.

SYNTAX

string **gui_zoom_to_selected_errors**

DESCRIPTION

If you have selected the errors in the error browser, you can execute this command to have layout zoom to the selected errors.

EXAMPLES

If you select errors, then perform layout zooming operations, the following command will force layout to zoom to selected errors:
gui_zoom_to_selected_errors

SEE ALSO

gui_error_browser(2)
gui_set_current_errors(2)
gui_set_selected_errors(2)
gui_set_error_browser_option(2)

help

Displays quick help for one or more commands.

SYNTAX

string **help**
  [-verbose]
  [-groups]
  [*pattern*]

Data Types

pattern string

ARGUMENTS

-verbose

Displays the command options; for example *command_name* **-help**.

-groups

Displays a list of command groups only.

pattern

Displays commands matching the specified pattern.

DESCRIPTION

The **help** command is used to get quick help for one or more commands or procedures. This is not the same as the **man** command that displays reference manual pages for a command. There are many levels of help.

By typing the **help** command, a brief informational message is printed followed by the available command groups.

List all of the commands in a group by typing the group name as the argument to **help**. Each command is followed by a one-line description of the command.

To get a one-line help description for a single command, type **help** followed by the command name. You can specify a wildcard pattern for the name; for example, all commands containing the string "alias". Use the **-verbose** option to get syntax help for one or more commands. Use the **-groups** option to show only the command groups in the application. This option cannot be combined with any other option.

EXAMPLES

The following example lists the commands by command group:

```
prompt> help
Specify a command name or wild card pattern to get help on individual
commands. Use the '-verbose' option to get detailed option help for a
command. Commands also provide help when the '-help' option is passed to
the command.'

You can also specify a command group to get the commands available in
that group. Available groups are:

 Procedures:          Miscellaneous procedures
 Help:                Help commands
 Builtins:            Generic Tcl commands
...
```

The following example displays the list of procedures in the Procedures group:

```
prompt> help procedures
 ls             # List files
 sh             # Execute a shell command
```

The following example uses a wildcard character to display a one-line description of all commands beginning with *a*:

```
prompt> help a*
 alias          # Create a command which expands to words.
 append           # Builtin
 array          # Builtin
```

This example displays option information for the **source** command:

```
prompt> help -verbose source
 source           # Read a file and execute it as a script
  [-echo]           (Echo all commands)
  [-verbose]        (Display intermediate results)
  file_name         (Script file to read)
```

SEE ALSO

man(2)
sh_help_shows_group_overview(3)

help_app_options

Displays the help information for application options.

SYNTAX

help_app_options
[-verbose]
[-scope *scope*]
[-category *category*]
[*patterns*]

Data Types

scope string
category string
patterns string

ARGUMENTS

-verbose

Displays the detailed information on the application options. This is an optional option. If this option is not specified, the information displayed is just the option name and the help string.

-scope *scope*

Specifies the scope of the application options for which the help string is to be displayed. This is an optional option. If this option is not specified, the help string for application options from both the global scope and design scope will be displayed.

-category *category*

Specifies the category of application options for which the help string is to be displayed. The help string for all the application options belonging to the specified category will be displayed.

patterns

Specifies a pattern for the option names. A pattern can include the wildcard characters asterisk (*) and question mark (?). "*" will match zero or more of any character, and "?" will match any single character. If this option is specified, the command displays the help string for only those application option names that match the specified pattern. If not specified, includes all the options.

DESCRIPTION

This command displays the help information for the application options whose names match the specified pattern.

The default behavior is to display the names of all the categories of registered application options.

EXAMPLES

The following example shows the default behavior

```
prompt> help_app_options
Categories of the registered application options are:
  abstract
  budget
  calc
  clock_opt
  constraint
  \...
  upf
  verilog
```

The following example displays the help string of all application options belonging to category 'verilog'.

```
prompt> help_app_options -category verilog
verilog.default_user_label          # Default user label to create the design in
verilog.enable_vpp                  # Enable Verilog pre-processor
verilog.keep_unconnected_cells      # Keep unconnected cells
verilog.keep_unconnected_nets       # Keep unconnected nets
```

The following example displays the help string for application options that are global scoped, belong to category 'shell' and match the pattern '*product*'

```
prompt> help_app_options -scope global -category shell *product*
shell.product_abbreviation          # Specifies the abbreviated string
for the product for this application.
shell.product_build_date            # Specifies the build date of this
application.
shell.product_name                  # Specifies the name for the product
for this application.
shell.product_version               # Specifies the version of this
application.
```

The following example displays a detailed information for the application option 'route.detail.max_antenna_pin_count'

```
prompt> help_app_options route.detail.max_antenna_pin_count -verbose
name                    route.detail.max_antenna_pin_count
value_type              integer
default_value           -1
help                    fix antenna on nets with number of
                        pins less than given value
status                  basic
minimum_value           -1
maximum_value           1000000
allowed_values          --
app_var_name            --
is_obsolete             false
is_global_only          false
```

```
    is_read_only                false
    is_persistent               true
    is_subscripted              false
```

SEE ALSO

set_app_options(2)
get_app_option_value(2)
get_app_options(2)
report_app_options(2)
reset_app_options(2)

help_attributes

Display help for attributes and object types

SYNTAX

string **help_attributes** [-verbose]
[-application]
[-user]
[*class_name*]
[*attr_pattern*]

string *class_name*
string *attr_pattern*

ARGUMENTS

-verbose

Display attribute properties.

-application

Only display application defined attributes.

-user

Only display user defined attributes.

class_name

Object class to retrieve help for.

attr_pattern

Show attributes matching pattern.

DESCRIPTION

The **help_attributes** command is used to get quick help for the set of available attributes. When this command is run with no arguments a brief informational message is printed followed by the available object classes.

EXAMPLES

```
prompt> help
cci_test> help_attributes
Specify an object type and an optional wild card pattern to get information
on the available attributes defined for the specified object type. Use the
-verbose option to get detailed information on the attributes

The available object types are:

 cell      net      pin
 ...
```

SEE ALSO

get_attribute(2)
help(2)
get_defined_attributes(2)

history

Displays or modifies the commands recorded in the history list.

SYNTAX

string **history**
  [-h]
  [-r]
  [*argument_list*]

Data Types

argument_list list

ARGUMENTS

-h

Displays the history list without the leading numbers. You can use this for creating scripts from existing history. You can then source the script with the **source** command. You can combine this option with only a single numeric argument. Note that this option is a nonstandard extension to Tcl.

-r

Reverses the order of output so that most recent history entries display first rather than the oldest entries first. You can combine this option with only a single numeric argument. Note that this option is a nonstandard extension to Tcl.

argument_list

Additional arguments to **history** (see DESCRIPTION).

DESCRIPTION

The **history** command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an "event." The most commonly used forms of the command are described below. You can combine each with either the **-h** or **-r** option, but not both.

- With no arguments, the **history** command returns a formatted string (intended for you to read) giving the event number and contents for each of the events in the history list.
- If a single, integer argument *count* is specified, only the most recent *count* events are returned. Note that this option is a

nonstandard extension to Tcl.

- Initially, 20 events are retained in the history list. You can change the length of the history list using the following:

  history keep *count*

Tcl supports many additional forms of the **history** command. See the "Advanced Tcl History" section below.

EXAMPLES

The following examples show the basic forms of the **history** command. The first is an example of how to limit the number of events shown using a single numeric argument:

```
prompt> history 3
   7  set base_name "my_file"
   8  set fname [format "%s.db" $base_name]
   9  history 3
```

Using the **-r** option creates the history listing in reverse order:

```
prompt> history -r 3
   9  history -r 3
   8  set fname [format "%s.db" $base_name]
   7  set base_name "my_file"
```

Using the **-h** option removes the leading numbers from each history line:

```
prompt> history -h 3
set base_name "my_file"
set fname [format "%s.db" $base_name]
history -h 3
```

Advanced Tcl History

The **history** command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an "event." When specifying an event to the **history** command, the following forms may be used:

- A number, which if positive, refers to the event with that number (all events are numbered starting at 1). If the number is negative, it selects an event relative to the current event; for example, **-1** refers to the previous event, **-2** to the one before that, and so on. Event **0** refers to the current event.
- A string selects the most recent event that matches the string. An event is considered to match the string either if the string is the same as the first characters of the event, or if the string matches the event in the sense of the **string match** command.

The **history** command can take any of the following forms:

history

Same as **history info**, described below.

history add *command [exec]*

Adds the *command* argument to the history list as a new event. If **exec** is specified (or abbreviated), the command is also executed and its result is returned. If **exec** is not specified, an empty string is returned.

history change *newValue [event_number]*

Replaces the value recorded for an event with *newValue*. The *event_number* specifies the event to replace, and defaults to the *current* event (not event **-1**). This command is intended for use in commands that implement new forms of history substitution and want to replace the current event (that invokes the substitution) with the command created through substitution. The return value is an empty string.

history clear

Erases the history list. The current keep limit is retained. The history event numbers are reset.

history event *[event_number]*

Returns the value of the event given by *event_number*. The default value of *event_number* is **-1**.

history info *[count]*

Returns a formatted string, giving the event number and contents for each of the events in the history list except the current event. If *count* is specified, then only the most recent *count* events are returned.

history keep *[count]*

Changes the size of the history list to *count* events. Initially, 20 events are retained in the history list. If *count* is not specified, the current keep limit is returned.

history nextid

Returns the number of the next event to be recorded in the history list. Use this for printing the event number in command-line prompts.

history redo *[event_number]*

Reruns the command indicated by *event* and returns its result. The default value of *event_number* is **-1**. This command results in history revision. See the following section for details.

History Revision

Pre-8.0 Tcl had a complex history revision mechanism. The current mechanism is more limited, and the **substitute** and **words** history operations have been removed. The **clear** operation was added.

The **redo** history option results in much simpler "history revision." When this option is invoked, the most recent event is modified to eliminate the history command and replace it with the result of the history command. If you want to redo an event without modifying the history, use the **event** operation to retrieve an event, and use the **add** operation to add it to history and execute it.

identify_channels

Identifies channels between specified objects. The supported objects are cells, site rows, placement blockages, routing blockages, and wires.

SYNTAX

status **identify_channels**
[-output_filename *file_name*]
[-horizontal_threshold *horizontal_threshold*]
[-vertical_threshold *vertical_threshold*]J
[-cross_area]
[-cross_area_width *cross_area_width*]
[-cross_area_height *cross_area_height*]
[*collection_of_objects*]

Data Types

file_name string
horizontal_threshold double
vertical_threshold double
cross_area_width double
cross_area_height double
collection_of_objects collection

ARGUMENTS

-output_filename *file_name*

Specifies the file name in which to write out the identified channels. By default, the command writes the channel information to "channels.txt".

-horizontal_threshold *horizontal_threshold*

Specifies the threshold value for horizontal channels. Channels taller than *horizontal_threshold* are not reported. If you specify **-horizontal_threshold** without specifying **-vertical_threshold**, only horizontal channels are reported.

-vertical_threshold *vertical_threshold*

Specifies the threshold value for vertical channels. Channels wider than *vertical_threshold* are not reported. If you specify **-vertical_threshold** without specifying **-horizontal_threshold**, only vertical channels are reported.

-cross_area

Identify cross area channels. Cross area channels have both a height and a width.

-cross_area_width *cross_area_width*

Specifies the width threshold value for cross area channels. Cross area channels with a width smaller than *cross_area_width* are not reported.

-cross_area_height *cross_area_height*

Specifies the height threshold value for cross area channels. Cross area channels with a height smaller than *cross_area_height* are not reported.

collection_of_objects

Specifies a collection of objects. The supported objects are cells, site rows, placement blockage, routing blockage, and wires.

DESCRIPTION

This command identifies channels between specified objects and writes the channel coordinates to channels.txt. The **-horizontal_threshold** and **-vertical_threshold** options specify an upper limit to the height and width of the channels; channels outside the limit are not reported.

The output format written to channels.txt of channel is

direction (xlo ylo) (xhi yhi)

For example,

vertical (1339.255 367.195) (1392.445 489.325)

EXAMPLES

The following example writes out a channel file named channels.txt for channels between cells SD_RFIFO_RAM and SD_WFIFO_RAM, site row unit_row_2, placement_blockage PB1, routing blockage RB1, and wires PATH_14_653.

```
prompt> set objects [get_cells "SD_RFIFO_RAM SD_WFIFO_RAM"]
prompt> append_to_collection objects [get_site_rows "unit_row_2"]
prompt> append_to_collection objects [get_placement_blockages "PB1"]
prompt> append_to_collection objects [get_routing_blockages "RB1"]
prompt> append_to_collection objects [get_shapes "PATH_14_653"]
prompt> identify_channels $objects
```

The following example writes out a channel file named channelObjects.txt and displays the file for channels between selected objects.

```
prompt> identify_channels [get_selection] -output_filename channelObjects.txt
prompt> sh cat channelObjects.txt
horizontal (1.6 1392.07) (352.165 1408.07)
horizontal (442.7 1408.07) (793.265 1424.07)
vertical (352.165 1424.07) (442.7 1616.515)
vertical (352.165 1648.515) (442.7 1856.96)
```

The following example writes out horizontal channels with a height less than 10 microns and vertical channels with a width less than 5 microns.

```
prompt> identify_channels [get_selection] -horizontal_threshold 10 -vertical_threshold 5
```

SEE ALSO

compile_pg(2)
create_pg_strap(2)

identify_multibit

Identifies multibit banking opportunities and writes out a register or mv_cell banking script.

SYNTAX

status **identify_multibit**
-register | -mv_cell
-output_file *file_name* | -apply
[-input_map_file *file_name*]
[-slack_threshold *slack_value*]
[-exclude_instance *exclude_cells*]
[-exclude_library_cells *library_cells*]
[-no_dft_opt]
[-cells *list_of_cells*]

Data Types

file_name string
slack_value float
exclude_cells list or collection
library_cells list or collection
list_of_cells list or collection

ARGUMENTS

-register

This option allows the user to identify groups of registers that can be replaced by multibit registers. You must specify either the **-register** or **-mv_cell** option.

-mv_cell

This option allows the user to identify groups of level_shifters or isolation cells that can be replaced by multibit level_shifter or isolation cell. You must specify either the **-register** or **-mv_cell** option.

-output_file *file_name*

Specifies the name of the output file that contains the generated banking commands. The output file contains a series of **create_multibit** commands. The output file can be sourced back into the tool to replace single-bit cells with multi-bit cells. You must specify either the **-output_file** or **-apply** option.

-apply

If this option is specified, tool modifies the netlist by replacing identified groups of single bit cells with multibit cells instead of

generating the output script file. You must specify either the **-output_file** or **-apply**.

-input_map_file *file_name*

Specifies the name of the input map file that contains functional group information and mapping information about which multibit register bank replaces which single-bit registers for each functional group.

If you do not specify this option, the command identifies the single-bit registers that can be replaced by available multibit registers in the libraries based on the functional information of the library cells; the tool then uses that information to control mapping.

This option can be used only when you are using the **-register** option. The format for the input map file is described in the FILE FORMATS section of this man page.

-exclude_instance *exclude_cells*

Specifies the list of instances to be ignored for grouping.

-exclude_library_cells *library_cells*

Specifies the list of library cells whose instances are ignored for grouping.

-slack_threshold *slack_value*

Specifies the slack value for a register to be considered for banking. Registers having slack value below this number is ignored for banking. By default tool consider all registers for banking. This option can be used only when you are using the **-register** option.

-no_dft_opt

This option disables the call to the **optimize_dft** command. If this option is not used, the netlist may get modified.

-cells *list_of_cells*

Specifies the list of single-bit cells, multibit cells or both to be considered for grouping.

DESCRIPTION

The **identify_multibit** command identifies groups of single-bit registers or mv_cells that can be replaced by multibit registers or mv_cells and then either modifies the netlist accordingly or generates a banking script file which can then be sourced back into the tool to replace single-bit cells with multibit cells. For high banking ratio consideration, **optimize_dft** is employed at the beginning by default, which may modify the netlist.

FILE FORMATS

Format for specifying -input_map_file

You can specify the input map file sequence as shown:

reg_group_name {*list_of_reference_of_single_bit_flops* }
bits { *number_of_instances ref_multibit_flop* } { ... }
bits { *number_of_instances ref_multibit_flop* } { ... }

Where *reg_group_name* specifies the name of the register group, *list_of_reference_of_single_bit_flops* specifies the list of references of single-bit registers. **bits** specifies the number of single-bit registers in a group to be replaced, *number_of_instance* specifies the number of multibit registers to be used, and *ref_multibit_flop* specifies the reference multibit library cell to be used.

For example:

```
reg_group_1 {REGX1 REGX2 REGX4}
2 {1 MREG2}
3 {1 MREG2}
4 {1 MREG4}
5 {1 MREG4}
6 {1 MREG2} {1 MREG4}
reg_group_2 {REGNX1 REGNX2 REGNX4}
2 {1 MREG2N}
3 {1 MREG2N}
4 {1 MREG4N}
5 {1 MREG4N}
6 {1 MREG2N} {1 MREG4N}
```

reg_group_1 {REGX1 REGX2 REGX4} means that single-bit registers whose reference is either REGX1, REGX2, or REGX4 can be grouped together. **4 {1 MREG4}** specifies that a group of four single-bit registers can be replaced by one cell whose reference is MREG4. Similarly, **6 {1 MREG2} {1 MREG4}** means a group of six single-bit registers can be replaced using one instances of MREG2 and one instance of MREG4.

You can include a comment using the # character in the input map file. Lines beginning with the # character to the end of the line is treated as a comment.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **identify_multibit** command to generate the banking script.

```
prompt> identify_multibit -register -input_map_file ./input.map -output_file ./create_bank.out
1
```

SEE ALSO

create_multibit(2)
split_multibit(2)

index_collection

Given a collection and an index into it, if the index is in range, create a new collection containing only the single object at the index in the base collection. The base collection remains unchanged.

Optionally, a second index can be passed which creates a new collection with the objects between the two indices in the base collection. Makes an implicit assumption that index<=index2

SYNTAX

collection **index_collection**
collection1
index
[index2]

Data Types

collection1 collection
index index
index2 index

ARGUMENTS

collection1

Specifies the collection to be searched.

index

Specifies the index into the collection.

The index is either a simple integer to specify the offset from the start of the collection or the string "end" optionally followed by a negative integer to specify the offset from the end of the collection.

For the start offset case the value must range from 0 to **sizeof_collection** - 1. For the end offset case the negative integer (if specified) must range from 0 to -(**sizeof_collection** - 1).

index2

Specifies an optional second index into the collection.

Specifies a second index to create a collection of a range of objects from the base collection, defined by the two indices [index, index2] (boundaries inclusive)

DESCRIPTION

You can use the **index_collection** command to extract a single object from a collection. The result is a new collection containing only that object. The index operation is done in constant time - it is independent of the number of elements in the collection, or the specific index.

The range of indices is from 0 to one less than the size of the collection. If the specified index is outside that range, an error message is generated.

Commands that create a collection of objects do not impose a specific order on the collection, but they do generate the objects in the same, predictable order each time. Applications that support the sorting of collections allow you to impose a specific order on a collection.

You can use the empty string for the *collection1* argument. However, by definition, any index into the empty collection is invalid. Therefore, using the **index_collection** command with the empty collection always generates the empty collection as a result and generates an error message.

Note that not all collections can be indexed.

EXAMPLES

The following examples from PrimeTime use the **index_collection** command to extract the first, second, last and penultimate object of a collection.

```
pt_shell> set c1 [get_cells {u1 u2 u3}]
{"u1", "u2", "u3"}
pt_shell> query_objects [index_collection $c1 0]
{"u1"}
pt_shell> query_objects [index_collection $c1 1]
{"u2"}
pt_shell> query_objects [index_collection $c1 0 1]
{"u1", "u2"}
pt_shell> query_objects [index_collection $c1 end]
{"u3"}
pt_shell> query_objects [index_collection $c1 end-1]
{"u2"}
pt_shell> query_objects [index_collection $c1 end-1 end]
{"u2", "u3"}
```

SEE ALSO

collections(2)
query_objects(2)
sizeof_collection(2)

infer_supply_from_pg_net

Infer supply net connections to leaf PG pins in power intent based on their existing net connections in the PG netlist.

SYNTAX

status **infer_supply_from_pg_net**
[-create_supply_port]
[-verbose]

ARGUMENTS

-create_supply_port

If specified, supply net connection translation into the power intent may entail creation of supply ports in certain specific cases such as when there are domain dependent supply nets or when connecting across scopes.

-verbose

Prints the details for the inference information for each pin.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

commit_upf(2)

infer_switching_activity

Infers the switching activity on the drivers of special control input pins in the current design.

SYNTAX

int **infer_switching_activity**
[-objects *object_list*]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-sci_based *type*]
[-apply]
[-verbose]
[-quiet]
[-output *file_name*]

Data Types

object_list list
mode_list list
corner_list list
scenario_list list
type string
file_name string

ARGUMENTS

-objects *object_list*

List of objects (pins, ports, nets, cells) for which to report the activity influencing essential points. For a net, the tool considers any essential point in its fanin. For pins or ports, the tool considers any essential point in the fanin; if the pin or port happens to be an essential point, it reports and applies activity for the pin/port itself. For cells, the tool considers any essential point that is in the fanin of its input pins.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering on modes occurs.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If this option is not specified, no filtering occurs on corners.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then the considers only those scenarios that contain the modes and corners specified through **-modes** and **-corners** options. The command output is generated separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options is specified, the command reports the current scenario in the design.

-sci_based ***type***

Specifies whether to report or apply activity on any essential point based on the number of special control inputs it influences. The **type** specified can take one of the following three values: **sci**, **non_sci**, and **all**. If type is **sci**, the activity for any essential point is reported and applied based on the number of SCIs (Special Control Inputs) in its fanout. If there is no SCI in the fanout of an essential point, the tool does not report for the point. For the **non_sci** type, the default activity value is reported and applied on all essential points that do not have any SCIs in their fanout. If the type is **all**, the tool reports and applies all the essential points. If this option is not specified, the default behavior corresponds to **all**.

-apply

Applies the proposed switching activity annotation on the drivers of the special control input pins. The activity type for points where activity has been applied is **inferred**.

-verbose

Reports detailed information for each special control input in the fanout of an object.

-quiet

Suppresses the output report of the command. If either **-apply** or **-output** option has been specified, activity will still be applied/written out to a file accordingly.

-output ***file_name***

Creates a file with **set_switching_activity** commands that can be used to apply the activity on all or specified essential points.

DESCRIPTION

This command infers the switching activity on the drivers of the Special Control Input pins (SCIs) in the design. These pins can be clock pins, set pins, reset pins, scan enables, isolation cell control pins, power switch control pins, and so on. The switching activity includes toggle rate and static probability on the drivers of these pins.

The **-objects** option can be used to filter the essential points for reporting and applying the activity. To filter for scenarios, the **-modes**, **-corners**, and **-scenarios** options can be used. To determine whether to report and apply activity for essential points with zero or nonzero SCI count, the **-sci_based** option can be used. If the **-sci_based** option is not specified, all essential points are considered for reporting and applying. Use the **-apply** option to apply the inferred switching activity values on the drivers of the pins. For a detailed report, the **-verbose** option can be used. The **-quiet** option can be used to suppress the output report. The **-output** option creates a file with **set_switching_activity** commands that can be used to annotate the inferred activity later.

EXAMPLES

The following example shows the report generated by the **infer_switching_activity** command.

```
prompt> infer_switching_activity
Information: Activity for scenario scenario1 was cached, no propagation required. (POW-005)
****************************************
```

```
Report : infer_switching_activity
Design : ChipTop
Version: O-2018.06-SP2-BETA
Date   : Wed Jun 20 01:30:31 2018
****************************************
Inferring activity for the scenario: scenario1
Mode        : func
Corner      : max_corner
Time Unit   : 1ns
Fastest clock: clk2 with period 1.0
--------------------------------------------------------------------------------
Object Type              : net
Object Name              : test_in1
Current Static Probability : 0.50
Current Toggle Rate      : 0.10
Proposed Static Probability: 0.00
Proposed Toggle Rate     : 0.00
Receiver Type: scan_enable | Receiver Count: 10
--------------------------------------------------------------------------------
```

In the following example, **infer_switching_activity** command is used with the *-verbose* option to get a more detailed report with activity information for individual SCIs. The *-apply* option applies the proposed activity on the object.

```
prompt> infer_switching_activity -verbose -apply
Information: Activity for scenario scenario1 was cached, no propagation required. (POW-005)
****************************************
Report : infer_switching_activity
       -verbose
Design : ChipTop
Version: O-2018.06-SP2-BETA
Date   : Wed Jun 20 01:36:06 2018
****************************************
Inferring activity for the scenario: scenario1
Mode        : func
Corner      : max_corner
Time Unit   : 1ns
Fastest clock: clk2 with period 1.0
--------------------------------------------------------------------------------
Object Type              : net
Object Name              : test_in1
Current Static Probability : 0.50
Current Toggle Rate      : 0.10
Proposed Static Probability: 0.00
Proposed Toggle Rate     : 0.00
           Receiver        Current          Current        Current
Receiver       Type    Activity Type   Static Probability     Toggle Rate

reg1/SE   scan_enable       default             0.50          0.10
reg2/SE   scan_enable       default             0.50          0.10
reg3/SE   scan_enable       default             0.50          0.10
reg4/SE   scan_enable       default             0.50          0.10
reg5/SE   scan_enable       default             0.50          0.10
reg6/SE   scan_enable       default             0.50          0.10
reg7/SE   scan_enable       default             0.50          0.10
reg8/SE   scan_enable       default             0.50          0.10
reg9/SE   scan_enable       default             0.50          0.10
reg10/SE  scan_enable       default             0.50          0.10
```

```
--------------------------------------------------------------------------------
```

In the following example, a specific object has been specified using the *-objects* option.

```
prompt> infer_switching_activity -objects [get_pins mux/S0]
Information: Activity for scenario scenario1 was cached, no propagation required. (POW-005)
****************************************
Report : infer_switching_activity
Design : ChipTop
Version: O-2018.06-SP2-BETA
Date   : Wed Jun 20 01:43:55 2018
****************************************
Inferring activity for the scenario: scenario1
Mode        : func
Corner      : max_corner
Time Unit   : 1ns
Fastest clock: clk2 with period 1.0
--------------------------------------------------------------------------------
Essential Points for pin 'mux/S0' :
Object Type              : port
Object Name              : clksel
Current Static Probability : 0.50
Current Toggle Rate        : 0.10
Proposed Static Probability: 0.00
Proposed Toggle Rate       : 0.00
Receiver Type: clock | Receiver Count: 9
Receiver Type: clock | Receiver Count: 1 (NEGATIVE)
--------------------------------------------------------------------------------
```

SEE ALSO

set_switching_activity(2)
report_switching_activity(2)
report_activity(2)
report_essential_points(2)
get_essential_points(2)

initialize_floorplan

Creates an initial floorplan with a die boundary, core, site array (or rows), and wire tracks. Support is provided for a die boundary that is coincident with the core, or a rectilinear core with rectangular die boundary.

SYNTAX

int **initialize_floorplan**
[-control_type core | die]
[-shape R | L | T | U]
[-side_length {side_a side_b [side_c side_d side_e side_f]}]
[-side_ratio {side_a side_b [side_c side_d side_e side_f]}]
[-core_utilization *ratio*]
[-keep_boundary]
[-boundary { {*x y*} {*x y*} {*x y*} {*x y*} ... }]
[-orientation N | W | S | E]
[-coincident_boundary true | false]
[-core_offset { *value* | *vertical_value horizontal_value* | *side_1 ... side_N*}]
[-row_core_ratio *ratio*]
[-flip_first_row true | false]
[-keep_pg_route]
[-keep_detail_route]
[-keep_placement {io macro block std_cell physical_only all}]
[-keep_objects *object_name_or_collection*]
[-keep_object_types {placement_blockage routing_blockage move_bound}]
[-keep_all]
[-honor_pad_limit]
[-site_def *site_def_name*]
[-use_site_row]
[-origin_offset {*x*, *y*}]
[-row_pattern {*row_pattern_name*}]

Data Types

ratio	float
x	float
y	float
value	float
vertical_value	float
horizontal_value	float
*side_**	float
object_name_or_collection	string or collection
site_def_name	string
row_pattern_name	string

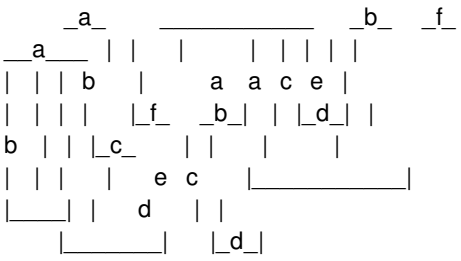
ARGUMENTS

-control_type core | die

Specifies whether the side_length and side_ratio options apply to the core or the die boundary. If set to die, then the dimensions will be applied to the die boundary and the core_offset values will be subtracted from the dimensions to determine the core boundary. If set to core (default), the dimensions will be applied to the core boundary and the core_offset values will be added to the dimensions to determine the final die boundary. By default, the control type is core.

-shape R | L | T | U

Specifies the shape to be used by the command. If the control_type is die, this option applies to the die boundary shape. The argument to this option specifies a template shape used to determine the cell boundary and core shape of the rectilinear block. The following diagram shows the definition of the edges and the orientation of the R-, L-, T-, and U- rectilinear blocks. By default, the core shape is R (rectangular).



-side_length { *side_a side_b side_c side_d side_e side_f*}

Specifies the side lengths for the edges of the floorplan. If the control_type is die, the side lengths apply to the die boundary. Each dimension in the list represents the length of the edge. If you provide more values than required to describe the specified shape, the extra values are ignored. If you do not provide all of the values required to describe the specified shape, the tool issues an error message. There are only two dimensions for **-shape R**: width and height. This option is mutually exclusive with the **-side_ratio** option.

-side_ratio { *side_a side_b side_c side_d side_e side_f*}

Specifies the relative proportion of the floorplan edges in relation to each other. If the control_type is die, the side rations apply to the boundary side settings. Each dimension in the list represents the relative proportion of the dimension of the edge to the sum of all the dimensions listed. For example, if the list of dimensions of an L-shaped block is {1 2 1 1}, the tool calculates the dimension of side a, c, or d (where the value is 1) as 20% (1/1+2+1+1) of the sum of the dimensions listed. The dimension of side b is 40% of the summation, and so on.

-core_utilization *ratio*

Specifies the utilization of the core area. The utilization is the total area of the core occupied by all standard cells and macro cells divided by the total core area. You can specify a value between 0 and 1. The cell area includes all standard and macro cells. For example, a core utilization of 0.8 specifies that 80 percent of the core area is used for cell placement at this stage. The tool might later add more cell area, the remaining area is available for routing. By default, the core utilization is 0.7.

-keep_boundary

Uses the existing die boundary. If this option is specified and the core-based constraints result in a core that is too large to fit in the existing die boundary, the command issues an error message. Default is not specified.

-boundary { {*x y*} {*x y*} {*x y*} {*x y*} ... }

Specifies the shape to be used by the command. If the control_type is core, then the boundary defines the core area and the

core_offsets should be added to create the die boundary. If control_type is die, then the core_offset is subtracted from the die boundary to create the core boundary.

-orientation N | W | S | E

Specifies one of four possible orientations for the specified rectilinear shape. The tool repositions the block to the specified orientation by rotating it in a clockwise direction. For **-shape R**, the orientation is always N.

-coincident_boundary true | false

Specifies whether the die boundary follows the shape of the core. If true, the die boundary assumes the same shape as the core and requires a **-core_offset** setting with the same number of sides as the core. If false, the die boundary is rectangular and the **-core_offset** option requires only four values. When the die boundary is rectangular it is created with **-core_offset** values such that the offset value is honored to the closest core edge on a per side basis. In this case, the bounding box of the die boundary is the minimum size that meets all four **-core_offset** values. By default, this option is true.

-core_offset { *value* | *vertical_value horizontal_value* | *side_1 ... side_N*}

Specifies the distance between the side of the core and the side of the die boundary. If only one value is specified, the value is used for all sides. If two values are specified, the first value is applied to all vertical edges and then second value is applied to all horizontal edges. Side numbers are based on the standard rectilinear numbering and do not correlate to the side_a, side_b, and so on. numbering scheme used to define the size of each edge. By default, the core offset equals to the minimum I/O cell height. If there are no I/O cells, the core offset is 0.

-row_core_ratio *ratio*

Specifies the amount of channel area between cell rows in the core area to reserve for routing. The *ratio* is a number between 0 and 1.0. A smaller row-to-core ratio creates more space for routing channels. A value of 1.0 creates no routing channel space. By default, the ratio is 1.0. Note that this ratio should be equal to or greater than the core utilization value.

-flip_first_row true | false

Specifies whether the command flips the first row at the bottom of the core area for horizontally placed cell rows, or flips the leftmost row for vertically placed cell rows. By default, this option is true.

-keep_pg_route

Specifies that the command keeps the PG routes and does not delete them. By default, the command delete all existing routes.

-keep_detail_route

Specifies that the command keeps all the routes except PG routes and does not delete them. By default, the command delete all existing routes.

-keep_placement {io macro block std_cell physical_only all}

Specifies that the command keeps the placement of specified object types. The valid value for this options are: *io, macro, block, std_cell, physical_only, all*.

io means objects with design type "*flip_chip_driver, flip_chip_pad, corner, pad, pad_spacer*";

macro means objects with design type "*macro, analog, abstract*";

block means objects with design type "*module, black_box*";

physical_only means objects with design type "*physical_only, fill*";

std_cell means objects with design type "*lib_cell, cover, diode, end_cap, well_tap, filler*";

all means all objects with design type list above.

-keep_objects *object_name_or_collection*

Specifies the objects to be kept. Currently supported objects are cells and nets. Specify the objects either by using an object access command, such as **get_cells** or **get_nets**, or by specifying object name patterns in a Tcl list.

-keep_object_types {placement_blockage routing_blockage move_bound}

Specifies the object type to be kept. Currently supported object types are placement_blockage, routing_blockage and move_bound.

-keep_all

Specifies this option to keep macro, std_cell, IO, physical_only, block, blockage, detail_route and pg_route.

-honor_pad_limit

Adjusts the core and die size to honor pad-limited designs. If this option is not specified, the core area is created based on the default core utilization ratio 0.7.

The option can be used only for rectangular floorplans, not for L, T, or U shapes.

The assumption for the command option is that the pad-type cells ()I/O pads or flip chip drivers) will be placed around the design boundary.

-site_def *site_def_name*

Specifies the site def to be used in floorplan when there are multiple site defs in the technology file. The default is to use default site def. If there is no default site def, the command uses the site def with the smallest site width.

-use_site_row

By default, the **initialize_floorplan** command creates siteArray objects. This option forces the command to create siteRow rather than siteArray.

-origin_offset {x y}

Specifies the location of the lower-left corner of the die boundary bbox with respect to the origin of the block.

-row_pattern {*row_pattern_name*}

Specifies the name of row_pattern to be used for floorplan when there are row patterns specification in the physical rule section of technology file.

DESCRIPTION

Creates a floorplan with a boundary, core, site array (or rows), and wire tracks. Before executing this command, you must open a physical design by using the **open_block** command, or create a design with the **read_verilog** or **read_verilog_outline** commands.

EXAMPLES

The following example creates a rectangular core and die boundary with a core utilization of 80% and a core offset of 1000 um for each side.

```
prompt> initialize_floorplan -core_utilization 0.8 \
    -core_offset {1000 1000 1000 1000}
```

The following example creates a rectangular die boundary and a T-shaped rectilinear core with the specified side dimensions for the core and a core_offset of 100 um for each side.

```
prompt> initialize_floorplan -control_type core -shape T \
        -side_length {1000 750 1500 750 1750 750} \
        -coincident_boundary false \
        -core_offset {100}
```

The following example creates a T-shaped rectilinear core and die boundary with the specified side dimensions for the core and a core offset of 100 um for each side.

```
prompt> initialize_floorplan -shape T \
        -side_length {1000 750 2500 500 3000 500} \
        -core_offset {100 100 100 100 100 100 100 100}
```

The following example creates a T-shaped rectilinear core and die boundary with the specified side ratios for the core and a core offset of 10 um for each side and a core utilization of 80%.

```
prompt> initialize_floorplan -core_utilization 0.8 \
        -shape T \
        -side_ratio {2 1 3 1 3 1} \
        -core_offset {100 100 100 100 100 100 100 100}
```

SEE ALSO

create_io_ring(2)
remove_io_rings(2)
report_io_rings(2)

insert_buffer

Adds buffer cells on the nets that are connected to the specified pins.

SYNTAX

collection **insert_buffer**
[-new_net_names *new_net_names*]
[-new_cell_names *new_cell_names*]
[-inverter_pair]
[-no_of_cells *number*]
object_list
[-lib_cell *buffer_lib_cell* | *buffer_lib_cell*]

Data Types

new_net_names list
new_cell_names list
number integer
buffer_lib_cell collection
object_list list

ARGUMENTS

-new_net_names *new_net_names*

Specifies the names of the new nets to add. You must specify one net name per buffer when adding buffers, and two net names per inverter pair when adding inverter pairs. If the specified net name already exists, the command adds a suffix of "_%d" or "_%d_%d" to the net name.

Optionally, if you specify only the common base name, the tool generates new net names by adding unique numeric suffixes to the common base name. The specified names can be any valid net names, but must be the leaf names. They must not be the hierarchical names and must not contain embedded hierarchical separators. They must be unique in the current context, as specified by the current instance. By default, the command uses the base name eco_net.

-new_cell_names *new_cell_names*

Specifies the names of the new cells to be added. You must specify one cell name per buffer when adding buffers, and two cell names per inverter pair when adding inverter pairs. If the specified cell name already exists, the command adds a suffix of "_%d" or "_%d_%d" to the cell name.

Optionally, if you specify only the common base name, the tool generates new cell names by adding unique numeric suffixes to the common base name. These names can be any valid cell names, but must be the leaf names. They must not be the hierarchical names and must not contain embedded hierarchical separators. They must be unique in the current context, as specified by the current instance.

By default, the command uses *eco_cell* as the common base name.

-inverter_pair

Adds inverter pairs instead of buffer cells. If you specify this option, you must supply a library cell that has an inverting output. You can use this option when the specified library cell or buffer has both inverting and noninverting outputs.

-no_of_cells ***number***

Specifies the number of buffer cells or inverter pairs to be inserted per net. The inserted repeaters are connected back-to-back in series. By default, the command inserts a single buffer cell or inverter pair per net.

object_list

Specifies a list of nets, pins, or ports that must be buffered. The new buffer cells or inverter pairs are placed close to the specified pins or ports if their cells are placed.

If you specify a net, the tool connects the buffers or inverter pairs such that a new buffer cell is the new load of the original net.

If you specify pins, the tool groups all of the specified pins based on the nets to which they are connected. When the grouped pins are load pins, the tool adds the buffers so that the new buffer cells can drive them. When the grouped pins are driver pins, the tool connects the new buffer cells so that they become the load of the specified driver pin.

-lib_cell ***buffer_lib_cell***

Specifies the library cell to be used as buffer. In this case, the object is either a named library cell or a library cell collection. This is a required option.

If the library cell has both inverting and noninverting outputs, that is, it can act both as buffer and inverter, the **-inverter_pair** option controls which output is used. If the library cell has multiple outputs, the command uses the first noninverting or inverting output.

buffer_lib_cell

This positional option is exactly equivalent to **-lib_cell**. And it is mutual exclusive with **-lib_cell**.

DESCRIPTION

This command adds buffers or inverter pairs at one or more specified pins or ports. A library cell with a single input and multiple outputs is a buffer, as long as each output has the same or inverted logic function as the input.

Like all other netlist editing commands, the command arguments to **insert_buffer** must be valid for the command to run successfully. If the command succeeds, the result is a collection of the newly-added cells. If the command fails, the command returns an empty collection or an empty string and does not change the netlist.

By default, each newly-created cell has a name beginning with the eco_cell string and ending with a unique numeric suffix. Each newly-created net has a name beginning with the eco_net string and ending with a unique numeric suffix. To override the automatic name generation by the tool, use the **-new_net_names** and **-new_cell_names** options.

You can mimic buffer addition by using other commands, such as **create_cell**, **create_net**, **disconnect_net**, and **connect_net**. The **insert_buffer** command provides a more efficient and safe way to add buffers.

This command support editing a verilog module in the module aspect, through **edit_module**.

This command honors design.eco_freeze_silicon_mode. When this app option is true, attribute is_fs_eco_add of buffers added will be true.

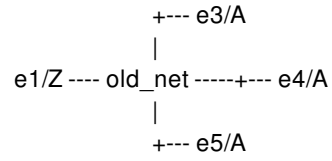
This command is exactly equivalent to **add_buffer**.

The **insert_buffer** command uses the following basic rules to check its arguments for validity:

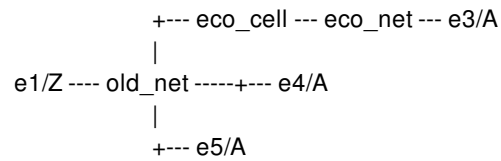
- The pin or port must be in scope; that is, at or below the current instance.
  For a description of special scoping rules, see the "Buffering Inside Boundary Pins" section.
- The pin or port must be connected to a net.
- Bidirectional pins cannot be buffered.
- The net cannot be either a PG net or a tie net.
- The specified library cell cannot be sequential.
- The specified library cell must be a buffer, as previously defined.
- The *number* argument of the **-no_of_cells** option must be a positive, nonzero integer.
- The library cell must have an inverting output, when you use the **-inverter_pair** option.
  The command uses the first inverting output and adds two cells so that it preserves the logic of the path.

Adding and Connecting the New Buffer

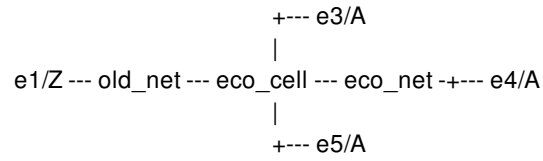
The following figure shows an example network that is used to describe the rules used by the **insert_buffer** command to connect the new buffer or inverter pair:



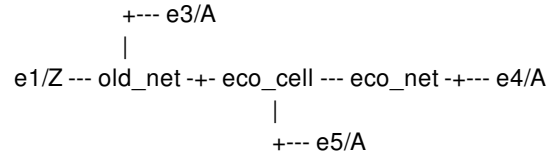
If the specified pin is a load, the load is first disconnected from its old net and then connected to the new net, as shown in the following figure:



If the specified pin is a driver, all loads on that net are disconnected from the old net and are connected to the new net, as shown in the following figure:



If you specify a list of load pins on the same net, these load pins are grouped and the new net drives them, as shown in the following figure:



Note that the tool cannot buffer a net driven by multiple drivers.

Buffering Inside Boundary Pins

When you use this command to add a buffer at a pin on the boundary of a hierarchical block, the command adds the buffer either inside or outside the hierarchical block, depending on the current hierarchical scope. To add the buffer inside the hierarchical block, set the scope to that block by using the **current_instance** command, and then run the **insert_buffer** command. The tool adds the buffer within the block to which the **current_instance** is set.

Don't touch support for nets

When app option *eco.add_buffer.honor_dont_touch* is set to true, nets with dont_touch attribute are skipped.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies a library cell that is not a buffer. The command fails and generates an error message.

```
prompt> insert_buffer e1/Z -lib_cell class/AN2P
Information: Buffering net old_net. (HFS-701)
Error: library cell 'AN2P' is not a buffer or an inverter. (ECOUI-010)
```

The following example specifies a buffer for the library cell:

```
prompt> insert_buffer e1/Z -lib_cell class/B1I
{eco_cell}
```

The following example specifies an inverter for the library cell. The command succeeds and creates the new cells named *eco_cell* and *eco_cell_1*. The new nets are named *eco_net* and *eco_net_1*. The **report_cells** command displays the connections for cell e3.

```
prompt> insert_buffer e3/A -lib_cell class/IV -inverter_pair
{eco_cell eco_cell_1}

prompt> report_cells -connections e3
****************************************
Report : cell
        -connections
****************************************

Connections for cell 'e3':
  Reference:        B1I
  Library:          class

  Input Pins        Net
  ----------------  ------------
  A                 eco_net

  Output Pins       Net
  ----------------  ------------
  Z                 out2
```

```
1
prompt> report_net -connections eco_net
****************************************
Report : net
       -connections
****************************************

Connections for net 'eco_net':

   Driver Pins          Type
   ------------         ----------------
   eco_cell/Z           Output Pin (IV)

   Load Pins            Type
   ------------         ----------------
   e3/A                 Input Pin (B1I)

1
```

Editing a verilog module in the module aspect:

```
prompt> edit_module r4000 { insert_buffer -lib_cell BUFFHVTD2 MemWrite -new_cell_names ecoCell -new_net_names ecoN
```

SEE ALSO

add_buffer(2)
current_instance(2)
get_pins(2)
remove_buffers(2)
report_cells(2)
report_nets(2)
size_cell(2)
edit_module(2)
eco.add_buffer.honor_dont_touch(3)

insert_dft

Inserts DFT structures in the current design

SYNTAX

status **insert_dft**

ARGUMENTS

This command has no arguments.

DESCRIPTION

insert_dft performs DFT structures insertion and routing. To insert scan compressor IP, clock controller or wrapper cells, use the applicable DFT configuration commands for that DFT logic type before running insert_dft.

EXAMPLES

prompt> insert_dft

SEE ALSO

set_dft_signal(2)
set_scan_compression_configuration(2)
set_scan_configuration(2)
set_core_wrapper_configuration(2)
set_pipeline_scan_data_configuration(2)
set_scan_path(2)
create_test_protocol(2)

insert_via_ladders

Inserts via ladders on pins with via ladder constraints.

SYNTAX

status **insert_via_ladders**
[-allow_drcs true | false]
[-allow_patching true | false]
[-allow_samenet_intersection true | false]
[-auto_stagger true | false]
[-bias_top_layer_to_samenet_connection true | false]
[-clean true | false]
[-connect_within_metal true | false]
[-connect_within_metal_for_via_ladder *mode*]
[-ignore_min_max_layer_constraints true | false]
[-ignore_rippable_shapes true | false]
[-ignore_routing_shape_drcs true | false]
[-ndr_mode full | top_layer_only | none]
[-ndr_on_top_layer_only true | false]
[-nets {*collection_of_nets*}]
[-pattern_must_join_over_pin_layer *number_of_levels*]
[-relax_line_end_via_enclosure_rule true | false]
[-relax_pin_layer_metal_spacing_rules true | false]
[-remove_routing_shapes_below_net_via_ladder_top_layer true | false]
[-shift_vias_on_transition_layers true | false]
[-strictly_honor_cut_table true | false]
[-top_layer_to_samenet_min_nonzero_distance *distance*]
[-user_debug true | false]
[-verbose true | false]

Data Types

mode string
{*collection_of_nets*} collection
number_of_levels int
distance float

ARGUMENTS

-allow_drcs true | false

Specifies whether via ladders are allowed to create DRC violations with routing shapes.

By default, this option is **false** and via ladders are inserted only if this can be done without creating any DRC violations.

If **true**, via ladder insertion tries to avoid creating DRC violations, but might create DRC violations with routing shapes if a clean solution cannot be found. Created DRC violations are tracked and reported by the command.

The recommended setting for this option is **false**. The via ladders that are created when this option is **true** might be structured to avoid DRC violations with routing shapes. Those routing shapes are often later ripped up and rerouted during detail routing. This option can therefore introduce a via ladder topology that is suboptimal and selected to avoid DRC violations with wires that might not be present after detail routing.

The recommended setting for optimal via ladder topology are **-allow_drcs false** and **-ignore_rippable_shapes true**, which is identical to **-ignore_routing_shape_drcs true**.

-allow_patching true | **false**

Specifies whether via ladder shapes can be patched to honor the minimum length rules specified in the technology file. The minimum length rules are specified by the **minLength** attribute in the Layer sections and The applicable minimum length on a layer is the maximum of these two values.

Generalized patching of other DRC types is not supported during via ladder insertion. However, the **upperMetalMinLengthTbl** attribute can sometimes be used to cover for other rule types.

The default is **false**.

-allow_samenet_intersection true | **false**

Specifies whether via ladder shapes on a net are allowed to intersect preroute shapes on the same net.

By default, this option is **false** and a via ladder is inserted on a pin but preroute shapes on the same net will be treated as blockages to be avoided. If the option is set to **true** a via ladder is allowed to touch other shapes on the same net as long as the result is DRC clean.

This option alone does NOT encourage connections to same net shapes. See the documentation for the **-bias_top_layer_to_samenet_connection** companion option.

-auto_stagger true | **false**

Specifies whether via ladder insertion is free to stagger by any amount on any layer to achieve a successful insertion.

The option only impacts attempts to insert via ladder template types that have no user-defined staggering values. If a template has user-defined values, the tool will honor those.

For example, auto stagger is only active for via ladder templates defined in the tech file with an empty or zeroed maxNumStaggerTracksTbl, or for Tcl-defined templates with empty or zeroed max_num_stagger_tracks_table. Staggering for other templates would occur normally, regardless of the option setting.

When auto stagger is active for an insertion attempt, the tool will analyze the track configurations and layout around the pin to determine if staggering should be enabled on a level of the via ladder and by how much. This relieves the need for a user to specify any staggering values in the template.

By default, this option is **false**.

-bias_top_layer_to_samenet_connection true | **false**

Specifies whether via ladder insertion should prefer to create a top level of a via ladder that intersects existing same net preroutes.

This option only has an impact when the **-allow_samenet_intersection** option is true and the top layer of a via ladder consists of 1 row.

If preconditions are met and this option is true, the tool always strongly prefers to create a via ladder top row so that it overlaps any existing samenet geometry on the top layer.

By default, this option is **false**.

-clean true | **false**

Specifies whether all existing via ladders in the design should be removed before proceeding with via ladder insertion.

By default, this option is **false** and existing via ladders are not removed. Insertion of a new via ladder is attempted only on pins that do not already have a via ladder.

-connect_within_metal true | **false**

Specifies whether via ladder shapes must be within pin shapes and their extensions on the pin layer.

This option controls both "is_via_ladder" and "is_pattern_must_join" shapes.

By default, this option is **false**.

-connect_within_metal_for_via_ladder ***mode***

Specifies whether via ladder shapes must be within pin shapes and their extensions on the pin layer.

This option controls only "is_via_ladder" shapes and overrides the setting of the **-connect_within_metal** option.

The default is **same_as_global**. The other two settings are **true** or **false**.

-ignore_min_max_layer_constraints true | **false**

Specifies whether minimum and maximum layer constraints are ignored during via ladder insertion.

By default, this option is **false**.

If **true**, minimum and maximum layer constraints will be ignored during via ladder insertion. Any "Min-max layer" DRC will not prevent successful insertion of a via ladder.

If **false** (default), a via ladder will not be inserted if minimum or maximum layer constraints would be violated.

-ignore_rippable_shapes true | **false**

Specifies whether detail routing shapes are ignored during via ladder insertion DRC checking.

By default, this option is **false**.

If **true**, via ladder insertion ignores any detail routing shape that the detail router would be allowed to reroute. DRCs are not checked against detail routing shapes during via ladder insertion.

-ignore_routing_shape_drcs true | **false**

This is a convenience option equivalent to using both **-allow_drcs false** and **-ignore_rippable_shapes true**, which are the recommended option settings for ideal via ladder topology.

By default, this option is **false**.

If **true**, any conflicting setting of the **-allow_drcs** and **-ignore_rippable_shapes** are ignored.

-ndr_mode full | **top_layer_only** | **none**

Controls how via ladders on nets with nondefault routing rules are created with default width wires.

If **full** (default), all layers of the via ladder will honor the NDR.

If **top_layer_only**, only the top layer of the via ladder will honor the NDR. All lower layers will use default width wires.

If **none**, the NDR is ignored and all layers of the via ladder will use default width wires.

This option replaces and overrides the **-ndr_on_top_layer_only** option. That legacy option will be obsoleted in a future release.

-ndr_on_top_layer_only true | false

Controls whether via ladders on nets with nondefault routing rules are created with default width wires.

By default, this option is **false** and the command honors the nondefault routing rules on all layers.

If **true**, the command honors the nondefault routing rules only on the top layer and uses default width wires on the lower layers.

-nets {*collection_of_nets*}

Specifies the nets on which to insert via ladders. A via ladder is inserted on pins with via ladder constraints only when the pin is also on one of the specified nets.

If you do not specify this option, via ladders are inserted on any pin with a via ladder constraint.

-pattern_must_join_over_pin_layer *number_of_levels*

Specifies number of levels of pattern-must-join structure. The default value is 1, and the maximum is 2.

- If the value set to 1, a single-level pattern-must-join structure is inserted.
- If the value set to 2, a single via is added on top of the original single-level pattern-must-join structure. The via on the second level has the **is_pattern_must_join** attribute and the router connects to it through its upper via enclosure.

-relax_line_end_via_enclosure_rule true | false

Specifies whether the command ignores the "End extension for via enclosure" rule during via ladder insertion.

By default, this option is **false**.

-relax_pin_layer_metal_spacing_rules true | false

Specifies whether the command can ignore some metal spacing rules on the pin layer during via ladder insertion whenever the involved via ladder shape is fully contained within the pin. This includes the "Diff net spacing," "End to end forbidden spacing," and "End to end space keepout" DRC types.

By default, this option is **false**.

-remove_routing_shapes_below_net_via_ladder_top_layer true | false

Specifies whether to remove all routing shapes below the maximum instantiated via ladder top layer on all nets with via ladders.

By default, this option is **false**.

If **true**, after inserting a via ladder on a net, the tool removes all routing shapes below the top layer of all inserted via ladders on that net.

-shift_vias_on_transition_layers true | false

Specifies whether via ladder cuts on transition layers can be shifted off the routing tracks for improved DRC compliance.

By default, this option is **false** and via ladder cuts are always centered at the intersection of routing tracks between adjacent layers.

If **true**, via ladder cuts can be shifted off track on transition layers for improved DRC compliance.

-strictly_honor_cut_table true | false

Specifies whether via ladder insertion must strictly honor the cuts specified in the via rule definition.

By default, this option is **false** and via ladder insertion usually honors the cuts specified in the via rule definition, but can choose

other cuts if they improve compliance with the fat via rules.

If **true**, via ladder insertion strictly honors the cut names specified in the via rule definition.

-top_layer_to_samenet_min_nonzero_distance ***distance***

This option only has an impact when the **-allow_samenet_intersection** option is true and bias_top_layer_to_samenet_connection is true.

See the documentation for the **-bias_top_layer_to_samenet_connection** option.

If a legal directly-overlapping top layer connection is not possible and a **-top_layer_to_samenet_min_nonzero_distance** value is specified, the tool tries to create a top layer row at least this distance from any samenet layout on the top layer. If this distance is not supplied or is 0, a row may be created any legal distance from samenet layout when a direct connection is impossible.

The distance is specified in user units.

If distance is supplied and a legal solution cannot be found, the tool tries to create any legal via ladder that satisfies all other constraints. I.e. the tool would create a via ladder as if the **-bias_top_layer_to_samenet_connection** option is false.

The default is **0**.

-user_debug true | **false**

Specifies whether to print additional information about via ladder insertion failures.

By default, this option is **false**.

If **true**, the command outputs information about track restrictions and flagged DRC types when via ladder insertion fails on a pin. This information can help you understand why via ladder insertion failed.

-verbose true | **false**

Species whether the command prints the success or failure status of each attempted via ladder insertion.

By default, this option is **false**.

DESCRIPTION

The **insert_via_ladders** command inserts via ladders on pins with via ladder constraints.

Via ladder rules are specified in a ViaRule section of the technology file or by using the **create_via_rule** command.

Via ladder constraints are assigned to pins by using the **set_via_ladder_constraints** or **set_via_ladder_rules** command.

EXAMPLES

The following example inserts a via ladder on any pin with a via ladder constraint. Non-fixed detail routing shapes are ignored. Patching is allowed. Extra information is printed for any via ladder insertion failure.

```
prompt> insert_via_ladders -ignore_routing_shape_drcs true \
  -allow_patching true -user_debug true
```

The following example removes all via ladders in the design and tries to create new via ladders.

```
prompt> insert_via_ladders -clean true -ignore_routing_shape_drcs true \
  -allow_patching true -user_debug true
```

The following example allows "is_via_ladder" shapes to extend beyond the pins, while confining "is_pattern_must_join" shapes within the pins.

```
prompt> insert_via_ladders -connect_within_metal true \
  -connect_within_metal_for_via_ladder false
```

SEE ALSO

refresh_via_ladders(2)
remove_via_ladder_constraints(2)
remove_via_ladders(2)
set_via_ladder_constraints(2)
set_via_ladder_rules(2)
verify_via_ladders(2)

is_false

Tests the value of a specified variable, and returns 1 if the value is 0 or the case-insensitive string **false**; returns 0 if the value is 1 or the case-insensitive string **true**.

SYNTAX

status **is_false**
value

Data Types

value string

ARGUMENTS

value

Specifies the name of the variable whose value is to be tested.

DESCRIPTION

This command tests the value of a specified variable, and returns 1 if the value is either 0 or the case-insensitive string **false**. The command returns 0 if the value is either 1 or the case-insensitive string **true**. Any value other than 1, 0, **true**, or **false** generates a Tcl error message.

The **is_false** command is used in writing scripts that test Boolean variables that can be set to 1, 0, or the case-insensitive strings **true** or **false**. When such variables are set to **true** or **false**, they cannot be tested in the negative in an **if** statement by simple variable substitution, because they do not constitute a true or false condition. The following example is not legal Tcl:

```
set x FALSE
if { !$x } {
  set y TRUE
}
```

This results in a Tcl error message, indicating that you cannot use a non-numeric string as the operand of "!". So, although you can test the positive condition, **is_false** allows you to test both conditions safely.

EXAMPLES

The following example shows the use of the **is_false** command:

```
prompt> set x TRUE
TRUE

prompt> if { ![is_false $x] } {
?       set y TRUE
?     }
TRUE

prompt>
```

SEE ALSO

expr(2)
if(2)
is_true(2)

is_true

Tests the value of a specified variable, and returns 1 if the value is 1 or the case-insensitive string **true**; returns 0 if the value is 0 or the case-insensitive string **false**.

SYNTAX

status **is_true**
value

Data Types

value string

ARGUMENTS

value

Specifies the name of the variable whose value is to be tested.

DESCRIPTION

This command tests the value of a specified variable, and returns 1 if the value is either 1 or the case-insensitive string **true**. The command returns 0 if the value is either 0 or the case-insensitive string **false**. Any value other than 1, 0, **true**, or **false** generates a Tcl error message.

The **is_true** command is used in writing scripts that test Boolean variables that can be set to 1, 0, or the case-insensitive strings **true** or **false**. When such variables are set to **true** or **false**, they cannot be tested in the negative in an **if** statement by simple variable substitution, because they do not constitute a true or false condition. The following example is not legal Tcl:

```
set x TRUE
if { !$x } {
  set y FALSE
}
```

This results in a Tcl error message, indicating that you cannot use a non-numeric string as the operand of "!". So, although you can test the positive condition, **is_true** allows you to test both conditions safely.

EXAMPLES

The following example shows the use of the **is_true** command:

```
prompt> set x FALSE
FALSE

prompt> if { ![is_true $x] } {
?       set y FALSE
?     }

FALSE

prompt>
```

SEE ALSO

expr(2)
if(2)
is_false(2)

legalize_placement

Executes detailed placement on the current design.

SYNTAX

status **legalize_placement**
[-cells *list_of_cells*]
[-moveable_distance *distance*]
[-boundary { {{llx lly} {urx ury}} | {{x1 y1} {x2 y2} ...} }]
[-incremental]
[-post_route]

Data Types

list_of_cells list or collection
distance float

ARGUMENTS

-cells *list_of_cells*

Specifies the cells to be legalized. You must set app-option 'place.legalize.legalize_only_selected_cells' to true or use command 'place_eco_cells -cells -legalize_only' before legalize_placement -cells. Without specifying the **-moveable_distance** option, only those cells will be legalized and all other cells not in the list will remain unchanged, no matter they are legal or not.

Example: To legalize all BUF* cells in the design

prompt> **set_app_option -name place.legalize.legalize_only_selected_cells -value true** prompt> **legalize_placement -cells [get_cells BUF*]** or prompt> **place_eco_cells -cells -legalize_only** prompt> **legalize_placement -cells [get_cells BUF*]**

Example: To legalize individual cells in the design

prompt> **set_app_option -name place.legalize.legalize_only_selected_cells -value true** prompt> **legalize_placement -cells "cellA cellB cellC"** or prompt> **place_eco_cells -cells -legalize_only** prompt> **legalize_placement -cells "cellA cellB cellC"**

-moveable_distance *distance*

By default, the value of moveable_distance is 0.0. That means only the specified cells will be legalized. When user specifies a moveable_distance value, other cells around that target cell will become moveable during legalization.

The size of moveable rectangle region based on the given distance will be {{llx - moveable_distance lly - moveable_distance} {urx + moveable_distance ury + moveable_distance}}. All the cells inside that rectangle region will be legalized and moveable while other cells not in the region will remain unchanged during legalization.

The moveable_distance value should be equal or larger than 0.0, and it can be used only with -cells option.

Example: To legalize all BUF* cells in the design, and include other cells around each BUF* cells within the specific distance

prompt> **legalize_placement -cells [get_cells BUF*] -moveable_distance 100**

To customize the moveable_distance value per cell instead of a global moveable_distance value, like high density area or critical timing area, user can create an user-defined attribute, and specify different moveable_distance value per cell. For example:

prompt> **define_user_attribute -type float -classes cell -name moveable_distance**

prompt> **set_attribute -name moveable_distance -value 1 [get_cells A]**

prompt> **set_attribute -name moveable_distance -value 100 [get_cells B]**

prompt> **set_attribute -name moveable_distance -value 10000 [get_cells C]**

prompt> **legalize_placement -cells "A B C" -moveable_distance 500**

So cell A will use 1 as its region value, cell B will use 100 as its region value and cell C will use 1000 as its region value. For other cells, they will use 500 as the region value.

The moveable distance will be considered during legalization only if user also specifies the **-cells** option.

-boundary { {{llx lly} {urx ury}} | {{x1 y1} {x2 y2} ...} }

Instead of specifying a collection of cells, user can use the **-boundary** option to specify a region of cells to legalize without knowing the cell name. The boundary syntax is {{llx lly} {urx ury}} or {{x1 y1} {x2 y2} ...}. All cells inside the specific region will be included and legalized while other cells will remain unchanged. Note that -boundary option includes cells based on cell's bbox, not based on cell's boundary bbox. In certain situations, cell's bbox may be bigger than cell's boundary bbox, so please make sure the -boundary region if it is very close to cells, user should check the cell's bbox values and adjust -boundary if needed.

Users may start the GUI to identify the area, get the coordinate information, and run legalization with the **-boundary** option to only legalize that specific area they have interest while other area will remain same. This option is similar to using the **-cells** option without the **-moveable_distance** option, but users don't have to know the all cell names.

The **-boundary** option should be exclusive with the **-cells** or **-moveable_distance** option.

Example: To legalize all cells inside the boundary without knowing their names

prompt> **legalize_placement -boundary {{100 100} {1500 1500}}**

-incremental

By using the **-incremental** option, **legalize_placement** first tries to find all illegal cells. The API used is same as for the **check_legality** command. Once it finds illegal cells, the **legalize_placement** command only legalizes those illegal cells while other legal cells will remain the same without any change. This is good idea when you don't know which cells are illegal, or you don't want to modify an existing design for those legal cells. The tradeoff is that when legalizer tries to fix those illegal cells, as legal cells will remain at the same location and cannot be moved, it may cause legalizer to move illegal cells to other available locations and cause larger displacement. You can use the **-cells** and **-moveable_distance** options to allow the legalizer to move some cells around those illegal cells to resolve larger displacement if this is needed.

The **-incremental** option should be exclusive with **-boundary** option.

-post_route

Specifies that post-route mode should be used. Legalizer will try to remain original orientation as possible.

DESCRIPTION

The **legalize_placement** command performs placement on the current design after coarse placement has already been performed.

When you run this command, the design should contain a floorplan with a site array specified and all cells and blockages in the design should have an initial location.

Multicorner-Multimode Support This command uses information from all active scenarios.

EXAMPLES

The following example runs the **legalize_placement** command:

```
prompt> legalize_placement
```

SEE ALSO

create_placement(2)

legalize_rp_groups

Executes detailed placement of relative placement groups in a design.

SYNTAX

status **legalize_rp_groups**
[*rp_groups*]
[-legalize_over_rp]

Data Types

rp_groups list or collection

ARGUMENTS

rp_groups

Specifies the relative placement groups for which you want to do detailed placement. If you do not specify this option, the command performs detailed placement on all relative placement groups of current block.

-legalize_over_rp

Controls the detailed placement of the specified relative placement groups. When this option is set, the cells of the specified relative placement groups can be legalized over the cells of unspecified relative placement groups.

-prototype

Enables faster legalization of the RP groups. If you specify this option then RP groups will be legalized much faster compared to when this option is not used. Specifying **-prototype** option disables few advanced rules resulting in faster search of legal location for cells.

DESCRIPTION

The **legalize_rp_groups** command performs placement legalization of relative placement groups in the current block, where coarse placement has already been performed. It is necessary to specify a site array for the floorplan. The legalization of relative placement groups of child blocks can not be done by this command.

Note: After you ran this command, the specified relative placement groups might overlap other unfixed cells. Therefore, you might need to run the **legalize_placement** command. If you do not need to perform detailed placement on the specified relative placement groups again, those relative placement groups can be fixed before you run the **legalize_placement** command. You can unfix them after running the **legalize_placement** command if necessary.

EXAMPLES

The following examples show how to use the **legalize_rp_groups** command.

prompt> **legalize_rp_groups rp_group**

prompt> **legalize_rp_groups rp_group -legalize_over_rp**

prompt> **legalize_rp_groups -prototype**

SEE ALSO

create_rp_group(2)
legalize_placement(2)
write_rp_groups(2)
create_placement(2)

link_block

Resolves block references.

SYNTAX

status **link_block**
[-force]
[-rebind | -incremental]
[-verbose]
[*block_name*]

Data Types

block_name string

ARGUMENTS

-force

Forces the block to relink, even if the block is already linked. If this option is not specified, the command does not link a block if it is already linked.

-rebind

Rebind all references using the current ref_lib list, view switch list and label switch list. The **-rebind** option is mutually exclusive with the **-incremental** option.

-incremental

Update only unresolved references. The **-incremental** option is mutually exclusive with the **-rebind** option.

-verbose

Writes out additional debugging messages.

block

Specifies the name of the block to be linked. If you do not specify this option, the command links the current block.

DESCRIPTION

This command performs a name-based resolution of block references for the specified *block_name* or for the current block.

References are resolved using other modules within this block, top modules of other physical hierarchy blocks, and leaf cells within lib cell libraries.

Libraries specified in the ref_lib list are then searched in order for references to physical hierarchy blocks and lib cells. The ref_lib list uses the *search_path* variable to locate libraries. The ref_lib list is usually set when the library is created with the **create_lib** command, but can be updated with the **set_ref_libs** command.

Physical hierarchy blocks can have multiple block views. The command determines the initial block view to use to resolve a reference based on the view switch list for the current block. If none of the selected views of a block exist, the reference is left unbound. Once bound to a particular view, a cell remains bound to that view unless rebound with **link_block -rebind**, **change_view**, or **change_abstract**. Use the **set_view_switch_list** command to set the view switch list for the session, a library, or a block.

Physical hierarchy blocks can also have multiple user labels. The reference resolves by default to the user label of the top-level block when label switch list is absent. For example, if the top-level block is created with the "netlist" label or unnamed ("") label, then all physical hierarchy block references are resolved using the respective label. If labels are absent, it will remain linked in its previous state. Note that lib-cell references unlike hierarchcical blocks, are only available and resolved under the default label. Once bound to a particular label, a cell remains bound to that label unless rebound with **link_block -rebind**. Use the **set_label_switch_list** command to set the label switch list for the session, a library, or a block.

Physical hierarchy blocks in a reference library can be linked using that library's ref_lib list by setting the *use_hier_ref_libs* attribute on that library to "true".

Unresolved References

If the linker fails to resolve one or more references, it creates an unbound leaf cell for each unresolved reference. To fix the problem, first determine and correct the source of the problem, then relink the block using the **link_block** command with the **-force** option. Failures are typically caused by missing libraries or blocks, an incorrectly specified ref_lib list, an incorrectly specified view switch list, label switch list or an incomplete *search_path* variable.

If you can resolve the references by changing the ref_lib list, view switch list, label switch list or *search_path* variable, you must relink the block with **link_block -rebind -force**.

You can limit the detailed reporting for unresolved references by setting the *link.reference_limit* app option. After this limit has been reached, the command reports only a summary of the number of additional unresolved references. Change *link.reference_limit* app option with the **set_app_options** command. Information on **link_block** app options can be found in the *link_options* manual page.

Link Errors

Mismatched port references to a lib cell or to a module in the same block cause the **link_block** command to fail. To correct the problem, you must remove the block, correct the Verilog netlist, and reread the design.

Circular or recursive module references cause the **link_block** command to fail. To correct the problem, you must remove the block, correct the Verilog netlist, and reread the design.

Auto Link

Many commands automatically link the current block before executing. These automatic links are always incremental, and might not fully rebind block references when the reference library list, view switch list or label switch list is changed. To force a full relink and rebinding of the block, use the **link_block -force -rebind** command.

EXAMPLES

In the following example, block TOP in library blockLib is linked to the physical hierarchy blocks MID and BOT, and the lib cell library

leafCellLib.

```
prompt> link_block TOP
Using libraries: designLib leafCellLib
Linking design TOP
Linking design BOT
Linking design MID
Information: units loaded from library 'leafCellLib' (LNK-040)
Design 'TOP' was successfully linked.
1
```

The following example shows the linking and relinking of views and labels.
The following commands set and get global view switch list.

```
prompt> set_view_switch_list {design abstract frame outline}
prompt> get_view_switch_list
design abstract frame outline
```

Similarly, the following commands set and get the global label switch list.

```
prompt> set_label_switch_list {label3 label2 label4 label1}
prompt> get_label_switch_list
label link map: { * {label3 label2 label4 label1 } }
```

Initial linking of views and labels is shown below :

```
prompt> current_design TOP
{leafCellLib:TOP.design}

prompt> link_block
Warning: Block 'leafCellLib:TOP.design' is already linked. (LNK-067)
0
prompt> get_blocks MID -hierarchical
{leafCellLib:MID.design}
```

The following command relinks and blocks are rebound to abstract view and label2 label.

```
prompt> set_view_switch_list {abstract frame design}
prompt> set_label_switch_list {label2 label3 label4 label1}
prompt> link_block -rebind
prompt> get_blocks MID -hierarchical
{leafCellLib:MID/label2.abstract}
```

The following command fails to rebind as the specified labels of block MID are absent.
The binding of block MID remains unchanged.

```
prompt> set_label_switch_list {label4 label1}
prompt> link_block -rebind
prompt> get_blocks MID -hierarchical
{leafCellLib:MID/label2.abstract}
```

For more info, see man page of these commands.

SEE ALSO

change_view(2)

change_abstract(2)
create_lib(2)
current_block(2)
current_design(2)
get_view_switch_list(2)
get_label_switch_list(2)
list_blocks(2)
read_verilog(2)
rebind_block(2)
remove_blocks(2)
set_app_options(2)
set_attribute(2)
set_ref_libs(2)
set_view_switch_list(2)
set_label_switch_list(2)
lib_attributes(3)
link_options(3)
search_path(3)

list_attributes

Lists currently defined attributes.

SYNTAX

string **list_attributes**
[-application]
[-class *class_name*]
[-nosplit]

class_name string

ARGUMENTS

-application

Lists application attributes as well as user-defined attributes.

-class *class_name*

Limit the listing to attributes of a single class.

To determine the valid classes, use the **get_defined_attributes -return_classes** command.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the output. Most of the information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **list_attributes** command displays an alphabetically sorted list of currently defined attributes. There are two categories of tool attributes: application-defined and user-defined. By default, **list_attributes** lists all user-defined attributes.

To add the application attributes to the listing, use the **-application** option. Note that there are many application attributes. It is often useful to limit the listing to a specific object class by using the **-class** option.

EXAMPLES

The following example lists the attributes defined with the **define_user_attribute** command.

```
prompt> list_attributes
****************************************
Report : List of Attribute Definitions
Version:
Date   : Wed Jun 15 19:22:34
****************************************

Properties:
   A - Application-defined
   U - User-defined
   I - Importable from design/library (for user-defined)
   S - Settable
   B - Subscripted

Attribute Name   Object  Type     Properties  Constraints
-------------------------------------------------------------
attr_b          cell    Boolean   U
attr_d          cell    double    U
attr_f          cell    float     U
attr_i          cell    int       U,I
attr_ir1        cell    int       U          0 to 100
attr_ir2        cell    int       U          >= 0
attr_ir3        cell    int       U          <= 100
attr_oo         cell    string    U          A, B, C, D
attr_s          cell    string    U
attr_s          net     string    U
```

The following example limits the listing to net attributes only.

```
prompt> list_attributes -application -class net
****************************************
Report : List of Attribute Definitions
Version:
Date   : Wed Jun 15 19:22:34
****************************************

Properties:
   A - Application-defined
   U - User-defined
   I - Importable from design/library (for user-defined)
   S - Settable
   B - Subscripted

Attribute Name        Object  Type     Properties  Constraints
-------------------------------------------------------------------
activity_type         net     string     A
antenna_rule_name     net     string     A,S
attr_s                net     string     U
base_name             net     string     A
bbox                  net     rect       A
connection_id         net     string     A
constraint_groups     net     collection A,B
 ...
user_dont_touch       net     boolean    A,S
voltage_range_max     net     float      A
```

voltage_range_min net float A

SEE ALSO

define_user_attribute(2)
get_attribute(2)
remove_attributes(2)
report_attributes(2)
set_attribute(2)

list_blocks

List blocks in one or more libraries.

SYNTAX

int **list_blocks** [-ref_libs] [-lib_cells] [*libraries*]

list *libraries*

ARGUMENTS

-ref_libs

List blocks from libraries in the current blocks's (or current library's) ref_lib list.

-lib_cells

Also list lib_cells within lib_cell libraries.

libraries

List blocks only from the given list of libraries.

-refresh

Refresh library catalog to bring in additional blocks added by another process.

DESCRIPTION

The **list_blocks** command lists the blocks from the specified libraries. By default, blocks from all libraries are listed. If a list of libraries are given, then only blocks from those libraries are listed. With the -ref_libs option, blocks are listed from the libraries in the order of the ref_libs for the current block. By default lib cells from lib cell libraries are not list. Lib cells can be listed with the -lib_cells option.

Each library is listed with its name and path. The current library is tagged "current". Libraries containing technology sections are tagged "tech". Lib cell libraries are tagged "lib_cell".

Each block in a library is listed with the block name and the block view type (abstract blackbox design frame outline timing). Each block has a count of the number of blocks containing instances that currently bound to this block.

Open blocks are marked with a "+", modified blocks are marked with a "*". Designs are tagged with their modification date or last saved date. The current block is tagged "current". Lib cell blocks are tagged "lib_cell".

Designs selected to resolve references by the **link_block** command are marked with a ">". This depends on the current block, the view switch list, and the current library's ref_lib list. See the **link_block** command for more information on linking blocks.

Many of these values can be accessed with the **get_attribute** command, see the **get_attribute** and **list_attributes** commands for more information on attributes.

EXAMPLES

This example lists the current blocks in libraries.

```
prompt> list_blocks
Lib refDesigns.nlib /user/bill/exportLibs/refDesigns.nlib tech
  +>  1 BOT.design Jul-23-09:29
  -   1 BOT.frame  Jul-23-10:14
Lib legacy.nlib /user/joe/mylibs/legacy.nlib tech
  -   1 BOT.design Aug-31-2011
  -   1 MID.design Sep-14-2011
  +   0 TOP.design Oct-03-2011
Lib myDesigns.nlib /user/joe/mylibs/myDesigns.nlib current
  +>  1 MID.design Jul-23-09:29
  *>  0 TOP.design Jul-23-09:29 current
7
```

This example lists the libraries in the current block's ref_lib list.

```
prompt> list_blocks -ref_libs
Lib myDesigns.nlib /user/joe/mylibs/myDesigns.nlib current
  +>  1 MID.design Jul-23-09:29
  *>  0 TOP.design Jul-23-09:29 current
Lib refDesigns.nlib /user/bill/exportLibs/refDesigns.nlib tech
  +>  1 BOT.design Jul-23-09:29
  -   1 BOT.frame  Jul-23-10:14
4
```

This example also lists the lib cell library.

```
prompt> list_blocks -ref_libs -lib_cells
Lib myDesigns.nlib /user/joe/mylibs/myDesigns.nlib current
  +>  1 MID.design Jul-23-09:29
  *>  0 TOP.design Jul-23-09:29 current
Lib refDesigns.nlib /user/bill/exportLibs/refDesigns.nlib tech
  +>  1 BOT.design Jul-23-09:29
  -   1 BOT.frame  Jul-23-10:14
Lib techlib /project/exportLibs/techlib.nlib lib_cell tech
  -   0 INV.design Jun-27-2011 lib_cell
  +   0 INV.frame  Jun-27-2011 lib_cell
  +>  3 INV.timing Jun-27-2011 lib_cell
  -   0 ND2.design Jun-27-2011 lib_cell
  +   0 ND2.frame  Jun-27-2011 lib_cell
  +>  3 ND2.timing Jun-27-2011 lib_cell
        ...
  -   0 MUX8.design Jun-27-2011 lib_cell
  +   0 MUX8.frame  Jun-27-2011 lib_cell
  +>  1 MUX8.timing Jun-27-2011 lib_cell
```

```
2254
```

This example lists blocks in the specified library.

```
prompt> list_blocks legacy.nlib
Lib legacy.nlib /user/joe/mylibs/legacy.nlib tech
  -  1 BOT.design Aug-31-2011
  -  1 MID.design Sep-14-2011
  +  0 TOP.design Oct-03-2011
3
```

SEE ALSO

current_design(2)
get_attribute(2)
get_blocks(2)
get_designs(2)
get_libs(2)
get_lib_cells(2)
get_view_switch_list(2)
link_block(2)
list_attributes(2)
open_block(2)
open_lib(2)
save_block(2)
save_lib(2)
set_ref_libs(2)
set_view_switch_list(2)

list_commands

Prints list of all the commands.

SYNTAX

status **list_commands**
[pattern]
[-status {*command_status*}]
[-options]
[-bg]

Data Types

pattern string
command_status string

ARGUMENTS

pattern

Specifies to print the commands matching this pattern only.

-status {*command_status*}

Specifies to print the commands of this status only.

Status can be one of normal, obsolete, deprecated, retired or all. Default value is all.

-options

Specifies to print the list of command options of the commands also.

-bg

Specifies to print the list of command supported within redirect -bg.

DESCRIPTION

The **list_commands** command will print names of all the commands and their attributes such as status (normal/deprecated/obsolete) and read-only/undoable by default.

When a pattern is specified after **list_commands**, only commands matching the pattern will be printed. Further the list of commands

can be filtered using *-status* option.

When *-options* is specified it will print the command options of the commands with their status (normal/deprecated).

EXAMPLES

The following example lists all the commands.

```
prompt> list_commands

Attributes:
N=Normal, D=Deprecated, O=Obsolete
R=Read-only, U=Undoable

Command Name             Attributes
------------             ----------
add_attachment           N
add_buffer               N, U
add_buffer_on_route      N, U
 ...
```

The following example lists all the commands with their command options and options attributes.

```
prompt> list_commands -options

Attributes:
N=Normal, D=Deprecated, O=Obsolete
R=Read-only, U=Undoable

Command Name          Attributes    Command Options    Options Attributes
------------          ----------    ---------------    ------------------
add_attachment        N             -file_name         N
                                    -object            N
                                    name               N

add_buffer            N, U          -inverter_pair     N
                                    -lib_cell          N
                                    -new_cell_names    N
                                    ...
 ...
```

The following example lists all the commands which match the pattern 'fi' with their command options and options attributes.

```
prompt> list_commands fi -options

Attributes:
N=Normal, D=Deprecated, O=Obsolete
R=Read-only, U=Undoable

Command Name          Attributes    Command Options    Options Attributes
------------          ----------    ---------------    ------------------
find_objects          N, R          -direction         N
```

```
                          -exact          N
                          -leaf_only      N
                          ...

   fix_signal_em      N          -nets          N
    ...
```

The following example lists all the commands which match the pattern 'map' and have status 'normal' with their command options and options attributes.

```
prompt> list_commands map -status normal -options

Attributes:
N=Normal, D=Deprecated, O=Obsolete
R=Read-only, U=Undoable

Command Name             Attributes    Command Options    Options Attributes
------------             ----------    ---------------    ------------------
map_freeze_silicon        N, U         -eco_cell          N
                                       -map_file          N
                                       -spare_cell        N
                                       ...

map_isolation_cell        N           -domain            N
                                       -lib_cells         N
                                       ...

map_level_shifter_cell    N           -domain            N
                                       -lib_cells         N
                                       ...
 ...
```

The following example lists all the commands supported within redirect -bg.

```
prompt> list_commands -bg

Commands allowed in redirect -bg are:
- all_clocks
- all_connected
 ...
```

list_licenses

Displays a list of licenses currently checked out by the user.

SYNTAX

status **list_licenses**

ARGUMENTS

The **list_licenses** command has no arguments.

DESCRIPTION

The **list_licenses** command lists the licenses you currently have checked out. If more than one copy of a license key is checked out, the number of keys checked out is shown in parentheses following the license name.

EXAMPLES

The following example shows the output from **list_licenses**:

```
prompt> list_licenses
Licenses in use:
        Fusion-Compiler-BE (2)
        Fusion-Compiler-AG (2)
1
```

SEE ALSO

check_license(2)
get_licenses(2)
remove_licenses(2)

list_test_models

Lists the designs in memory that have CTL test models attached to them.

SYNTAX

status **list_test_models**

ARGUMENTS

The **list_test_models** command has no arguments.

DESCRIPTION

This command displays the designs in memory that have CTL test models attached to them. Designs that do not have CTL test model information are not listed. All designs in memory are analyzed, even if they are not a part of the current design or its hierarchy.

EXAMPLES

The following example lists the designs that have CTL test model information:

```
prompt> list_test_models
  des_unit                  /remote/my_dir/des_unit_xtol/db/des_unit.db
  des_unit_SCCOMP_COMPRESSOR     /remote/my_dir/des_unit_xtol/ \
                        des_unit_SCCOMP_COMPRESSOR.db
  des_unit_SCCOMP_DECOMPRESSOR   /remote/my_dir/des_unit_xtol/ \
                        des_unit_SCCOMP_DECOMPRESSOR.db
```

list_test_modes

Displays all of the test modes that are defined for the current design.

SYNTAX

status **list_test_modes**

ARGUMENTS

The **list_test_modes** command has no arguments.

DESCRIPTION

The **list_test_modes** command displays all of the modes defined for the current design. The **define_test_mode** command is used to define a mode of operation for a design.

If a design does not have any test modes defined, it displays the message "Design has no test modes."

EXAMPLES

The following is an example of the test modes report:

```
prompt> list_test_modes
Test Modes
==========

    Name: ScanCompression_mode
    Type: InternalTest
    Focus:
    Core des_unit_U_decompressor in decompress mode
    Core des_unit_U_compressor in compress mode

    Name: Internal_scan
    Type: InternalTest
    Focus:
```

```
        Name: Mission_mode
        Type: Normal
    1
```

SEE ALSO

current_design(2)
current_test_mode(2)
define_test_mode(2)

lminus

Removes one or more named elements from a list and returns a new list.

SYNTAX

list **lminus**
[-exact]
original_list
elements

Data Types

original_list list
elements list

ARGUMENTS

-exact

Specifies that the exact pattern is to be matched. By default, **lminus** uses the default match mode of **lsearch**, the **-glob** mode.

original_list

Specifies the list to copy and modify.

elements

Specifies a list of elements to remove from *original_list*.

DESCRIPTION

The **lminus** command removes elements from a list by using the element itself, rather than the index of the element in the list (as in **lreplace**). The **lminus** command uses the **lsearch** and **lreplace** commands to find the elements and replace them with nothing.

If none of the elements are found, a copy of *original_list* is returned.

The **lminus** command is often used in the translation of Design Compiler scripts that use the subtraction operator (-) to remove elements from a list.

EXAMPLES

The following example shows the use of the **lminus** command. Notice that no error message is issued if a specified element is not in the list.

```
prompt> set l1 {a b c}
a b c

prompt> set l2 [lminus $l1 {a b d}]
c

prompt> set l3 [lminus $l1 d]
a b c
```

The following example illustrates the use of **lminus** with the **-exact** option:

```
prompt> set l1 {a a[1] a* b[1] b c}
a a[1] a* b[1] b c

prompt> set l2 [lminus $l1 a*]
{b[1]} b c

prompt> set l3 [lminus -exact $l1 a*]
a {a[1]} {b[1]} b c

prompt> set l4 [lminus -exact $l1 {a[1] b[1]} ]
a a* b c
```

SEE ALSO

lreplace(2)
lsearch(2)

load_block_constraints

Loads constraints for child and/or top blocks.

SYNTAX

status **load_block_constraints**
-type PRECHECK | DEF | UPF | SDC | BUDGET | CLKNET | CTS_CONSTRAINT | BTM | FLOORPLAN | SCANDEF
-all_blocks | -blocks *block_designs*
[-host_options *host_option_name*]
[-work_dir *directory_name*]

Data Types

block_designs list
host_option_name string
directory_name string

ARGUMENTS

-type PRECHECK | DEF | UPF | SDC | BUDGET | CLKNET | CTS_CONSTRAINT | BTM | FLOORPLAN | SCANDEF

Specifies the type(s) of constraints to load or check. Valid values are: PRECHECK, DEF, UPF, SDC, BUDGET, CLKNET, CTS_CONSTRAINT, BTM, FLOORPLAN, SCANDEF. The values are case-insensitive.

The types, except *PRECHECK*, correspond to the keywords used in the constraint mapping file set through **set_constraint_mapping_file** command.

When type *PRECHECK* is specified, the tool prints information about the physical hierarchy tree of the design. If any other type is specified together with type *PRECHECK*, the information in the constraint mapping file related to the specified type is also reported.

When the command is issued without type *PRECHECK*, the tool loads the constraint of the specified types to the specified blocks according to information in the constraint mapping file.

You can specify *-type* multiple times with one **load_block_constraints** command.

-all_blocks

Loads constraints for all the child blocks in the current design and the current design itself, if the specified constraint types for the block are available in the constraint mapping file.

In a multi-level physical hierarchy design, the tool determines the nesting order of the blocks and runs the blocks in a bottom-up order.

This option is mutually exclusive with *-blocks*. One and only one of them should be specified.

-blocks *block_designs*

Specifies the list of child block designs to load constraints to.

In a multi-level physical hierarchy design, the tool determines the nesting order of the blocks and runs the blocks in a bottom-up order.

This option is mutually exclusive with *-all_blocks*. One and only one of them should be specified.

-host_options *host_option_name*

Specifies the host option to use for distributed processing. If not specified, the tool uses the global host option if the option contains a distributed processing settings.

-work_dir *directory_name*

Writes all generated scripts and log files to the specified directory.

By default, the work directory is *./work_dir*. If there are host option settings for distributed processing, the work_dir setting from the host option is used. This option takes precedence over the default and the host option setting.

DESCRIPTION

This command loads constraints for child blocks as well as for top level.

The tool gets the constraint files from the constraint mapping file. See the man page for **set_constraint_mapping_file** for details of how to set up the constraint mapping file.

DEF constraints are loaded through command **read_def**.

UPF constraints are loaded through command **load_upf** and only when UPF constraints has not been loaded previously for a given block. After **load_upf** is run, **commit_upf** is performed.

SDC constraints are loaded through command **source**. Operations **remove_modes -all** and **remove_corners -all** are performed before loading the SDC constraints for a block.

BTM stands for *blackbox timing model*. BTM constraints are loaded using command **source**.

FLOORPLAN, CLKNET and BUDGET constraints are loaded through command **source**.

CTS_CONSTRAINT constraints are loaded through command **source**. Then **synthesize_clock_trunks** is performed on the block.

Only blocks whose specified constraint types that are available in the constraint mapping file are run. If the abstract view exists for a block, the same type of abstract would be re-created after loading the constraints for that block.

If a block started as an outline view, command **expand_outline** is first performed on the block. Then constraints are loaded for the block and the abstract view is created. Any necessary **change_abstract** commands are added for any parent block where it is needed.

In a multi-level physical hierarchy design, the tool determines the nesting order of the blocks and run the blocks in a bottom-up order.

EXAMPLES

The following example UPF and SDC for all the child blocks as well as the top level are loaded.

```
prompt> set_constraint_mapping_file ./split/mapfile
prompt> load_block_constraints -all_blocks -type SDC -type UPF
```

The following example shows that FLOORPLAN constraints are loaded to all the child blocks as well as the top level.

```
prompt> set_constraint_mapping_file ./floorplan/mapfile
prompt> load_block_constraints -all_blocks -type FLOORPLAN
```

The following example CTS constraints are loaded to the block BOT and **synthesize_clock_trunks** is performed on BLK_A.

```
prompt> load_block_constraints -blocks BLK_A -type cts_constraint
```

The following example prints information about the design's physical hierarchy.

```
prompt> load_block_constraints -type precheck
```

SEE ALSO

commit_blackbox_timing(2)
report_constraint_mapping_file(2)
set_constraint_mapping_file(2)
write_floorplan(2)

load_busplans

Loads XML data and re-creates busplans.

SYNTAX

status **load_busplans**
-xml_file *filename*
[-only_create]
[-replace_existing]
[-verbose]
[-ignore_mib_compliance]
[-buses *patterns*]

Data Types

filename string
patterns string (patterns)

ARGUMENTS

-xml_file *filename*

Specifies the name of the XML data file to read. The XML file should have the same format as written by the *write_busplans -xml_file filename* command.

-only_create

Creates only the start and end parts of the busplans in the data file. No virtual registers or pins will be re-inserted into the busplans.

-replace_existing

Remove the existing busplan, specified by the **-buses patterns** argument, before re-creating it. Matching is done only based on busplan name; no other checking is done. If not specified, busplans that already exist are skipped.

-verbose

Prints more detail about each busplan as it is being re-created. Messages include errors detected during the process. Otherwise, the command prints only a summary at the end.

-ignore_mib_compliance

Do not try to re-create busplans in the order required for MIB compliance. As well, if any required MIB parent busplans are not in the explicit list of busplans to re-create, do NOT add them to the list automatically.

By default, MIB parent busplans will be created first, even if they appear later in the XML file, or are not in the explicit list of

busplans to re-create.

-buses *patterns*

Specifies the names of busplans to re-create, if found inside the XML data file. Pattern matching is done based on the Tcl "string match" algorithm. Multiple patterns can be included in one Tcl list such as: -buses {busABC* busDEF* bus3}.

DESCRIPTION

Loads and re-creates busplans from an XML data file, such as one previously written by *write_busplans -xml_file filename*.

EXAMPLES

The following example loads a busplan from the XML-formatted file named busplan.xml

prompt> **load_busplans -xml_file busplan.xml**

SEE ALSO

create_busplans(2)
get_busplans(2)
modify_busplan(2)
report_busplans(2)
write_busplans(2)

load_metal_pattern_density

Reads the metal density information of hard macro, including layer, bounding box and metal density value.

SYNTAX

int **load_metal_pattern_density**

Data Types

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **load_metal_pattern_density** command reads the metal density information for hard macros. The information includes layer, bounding box and metal density value. Using the command in the GUI, you can configure the visual metal pattern density map.

EXAMPLES

The following example loads the metal density information for the macro.

```
prompt> load_metal_pattern_density
1
```

SEE ALSO

load_of

Gets the capacitance of a library cell pin.

SYNTAX

float **load_of**
lib_pin

Data Types

lib_pin collection

ARGUMENTS

lib_pin

Specifies the name of the library cell pin, or a collection that contains the library cell pin, for which to get the capacitance. But the tool wrill only return the capacitance for first lib pin in the collection.

DESCRIPTION

The **load_of** command returns the capacitance of the first given library cell pin.

load_upf

Reads in a script in Unified Power Format (UPF) format.

SYNTAX

status **load_upf**
[-scope *instance_name*]
[-noecho]
[-strict_check true | false]
[-supplemental *supf_file_name*]
file_name

Data Types

instance_name string
file_name string

ARGUMENTS

-scope *instance_name*

Specifies the scope where the UPF commands contained in the *file_name* file are to be executed.

If you do not use this option, the command uses the current scope.

-noecho

If specified, commands in the UPF file are not echoed to the output.

-strict_check true | false

Specifies whether to perform strict name matching during execution of the UPF commands. Use this option only in the golden UPF mode.

Set this option to **true** when you load the golden UPF file into the synthesis tool for the very first time, when object names in the UPF commands are expected to match object names in the design exactly.

Set this option to **false** when you reapply the golden UPF file any time after the tool has operated on the design, when object names in the file are not expected to match object names in the design due to optimization, grouping, and wildcard name expansion. In this case, the tool identifies objects using rule-based matching and name map file, if any.

In front-end mode, the default setting is **true**; if you do not use the **-strict_check** option in the golden UPF mode, the tool uses strict name checking.

In back-end mode, the default setting is **false**; if you do not use the **-strict_check** option in the golden UPF mode, the tool uses non-strict name checking.

-supplemental *supf_file_name*

Specifies the name of the supplemental UPF file to be read. This option is supported only in the golden UPF mode.

When loading a UPF file with this option, any UPF that is not explicitly preceded by "set derived_upf false" is designated as "set derived_upf true". Supplemental UPF content is always loaded with the **-strict_check true** setting.

file_name

Specifies the name of the file that contains the UPF script to be read.

DESCRIPTION

The **load_upf** command sets the scope to the specified instance and executes the set of UPF commands in the specified file. When the command completes, it restores the current scope to what it was before. The commands are executed in the same way as the Tcl **source** command.

The tool automatically identifies the UPF constraint type based on the setting of the Synopsys-specific **derived_upf** variable, which is set on the design UPF by the Synopsys implementation tools when they update the UPF constraints, such as by adding **connect_supply_net** constraints. When the **derived_upf** variable is **true**, the constraints loaded by the **load_upf** command are considered supplemental constraints. When this variable is **false**, the constraints loaded by the **load_upf** command are considered golden constraints.

The following example shows a UPF file that has been modified by the tool:

```
## Constraints below are generated by icc2. Please do not modify.
set derived_upf true
if ($derived_upf) {
  connect_supply_net VDD_VSS.power -ports {
    u0_0/U41/VDD \
    u0_0/U24/VDD u0_0/U4/VDD \
    u0_1/U321/VDD }
}
set derived_upf false
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example loads the UPF script file named top.upf.

```
prompt> load_upf top.upf
1
```

SEE ALSO

source(2)
mv.upf.enable_golden_upf(3)
read_name_map(2)

log_trace

Control creation of a replay log of traced commands.

SYNTAX

string **log_trace** [-start file_name|-stop|-pause|-resume] [-log log_string] [-quiet]

string *file_name*
string *log_string*

ARGUMENTS

-start *file-Name*

This option is mutually exclusive with -stop, -pause, and -resume. The argument is the pathname of the file to which the replay trace log is output. The option opens a file for output and begins logging command execution strings and application variable changes. Note that if the file already exists, the command overwrites the existing file. There can only be a single replay trace log output at any time.

-stop

This option is mutually exclusive with -start, -pause, and -resume. This option stops replay trace log output and closes the log file. Once stopped, the replay log cannot be repopened for appending.

-pause

This option is mutually exclusive with -start, -stop, and -resume. This option pauses replay trace log output. Execution of all commands and application variable settings are ignored and omitted from the log until output is resumed with -resume.

Note that pausing replay log output may render the resulting log unable to replay or incapable of replicating tool results.

-resume

This option is mutually exclusive with -start, -stop, and -pause. This option resumes replay trace log output.

-log *log_string*

This option outputs the given log_string, verbatim, to the replay trace log. The option may be used to explicitly log a command invocation or a comment. If the string is a comment, it must start with the Tcl comment character '#'.

-quiet

If given, this option causes the command to ignore error conditions such as an attempt to pause the trace log output before it is started. The command will exit quietly when it encounters an error condition.

DESCRIPTION

The **log_trace** command performs operations related to creating a flat replay log of traced command execution and application variable changes. All options are optional. Upon successful completion, the command returns the pathname of the trace log file, if any, or an error message if the command fails.

Trace log output contains a comment header with the tool name and version, and the date and time when trace log output was started.

The resulting replay log is not 100% guaranteed to be able to replay or to reproduce tool results. One reason is that the tool execution environment may be different. If any operations are performed during log creation that would further compromise the ability to replay the resulting log or to replicate tool results, a warning message is appended to the log. Such operations including pausing log output.

Invoking the **log_trace** with no options will report the current status of logging: on, paused, or off, followed by the currently open log file name, if any.

EXAMPLES

The following example shows reporting of the log status before starting trace logging.

```
prompt> log_trace
off
```

The following example starts trace log creation.

```
prompt> log_trace -start ./tracelog.tcl
/an/absolute/path/.tracelog.tcl
prompt> log_trace
on: /an/absolute/path/.tracelog.tcl
```

The following example pauses, then resumes log creation.

```
prompt> log_trace -pause
/an/absolute/path/.tracelog.tcl
prompt> log_trace
paused: /an/absolute/path/.tracelog.tcl
prompt> log_trace -resume
/an/absolute/path/.tracelog.tcl
prompt> log_trace
on: /an/absolute/path/.tracelog.tcl
```

SEE ALSO

set_trace_option(2)
get_trace_option(2)
annotate_trace(2)

ls

Lists the contents of a directory.

SYNTAX

string **ls** [*filename ...*]

string *filename*

ARGUMENTS

filename

Provides the name of a directory or filename, or a pattern which matches files or directories.

DESCRIPTION

If no argument is specified, the contents of the current directory are listed. For each *filename* matching a directory, **ls** lists the contents of that directory. If *filename* matches a file name, the file name is listed.

EXAMPLES

shell> **ls *.db *.pt**

test1.pt c1.db c3.db c5.db

test2.pt c2.db c4.db c6.db

SEE ALSO

cd(2)
pwd(2)

magnet_placement

Performs magnet placement on current design.

SYNTAX

status **magnet_placement**
[-move_fixed]
[-mark_fixed]
[-move_legalize_only]
[-mark_legalize_only]
[-stop_by_sequential_cells]
[-stop_on_sequential_cells]
[-exclude_buffers]
[-avoid_soft_blockage]
[-only_report_magnet_cells]
[-get_collection]
[-logical_levels *level*]
[-multiple_long_port_mode auto | nearby]
[-cells *object_list*]
[-stop_points *object_list*]
[-hierarchy_mode default | block | all]
magnet_objects

Data Types

level integer
object_list collection
magnet_objects collection

ARGUMENTS

-move_fixed

Specifies that the cell marked as fixed can be moved. By default, the fixed cells and the cells behind them won't be pulled.

-mark_fixed

Specifies that the cells are to be marked fixed after running the **magnet_placement** command. By default, no cell is marked as fixed after the command. This option is mutually exclusive with **-mark_legalize_only**.

-move_legalize_only

Specifies that a cell marked as legalize_only can be moved. By default, legalize_only cells cannot be moved.

-mark_legalize_only

Specifies that the cells are to be marked as legalize_only after running the **magnet_placement** command. By default, no cell is marked as legalize_only after the command. This option is mutually exclusive with **-mark_fixed**.

-stop_by_sequential_cells

Stops pulling cells when a sequential cell is encountered during level traversal. The sequential cell is not pulled itself. This option is mutually exclusive with **-cells**, **-stop_on_sequential_cells** and **-stop_points**.

By default, cells in logical level 1 only are considered for pulling. Logical level can be changed using **-logical_level** command option.

-stop_on_sequential_cells

Stops pulling cells when a sequential cell is encountered during level traversal. The sequential cell is pulled itself. This option is mutually exclusive with **-cells**, **-stop_by_sequential_cells** and **-stop_points**.

By default, cells in logical level 1 only are considered for pulling. Logical level can be changed using **-logical_level** command option.

-logical_levels *level*

Specifies the number of logical levels to pull. By default, *level* is 1 and only the first-level cells are pulled. This option is mutually exclusive with **-cells**.

-cells *object_list*

Pulls only the cells in the specified object list. You specify only cell objects in the object list. This option is mutually exclusive with **-logical_levels** and **-stop_by_sequential_cells**.

The cells that you specify should form an intact datapath with magnet object. For example, if magnet object connects to cell_a, and cell_a connects to cell_b, you should specify both cell_a and cell_b. These cells do not have to be a timing path.

-exclude_buffers

Ignores buffers and inverters when calculating the level. Buffers and inverters are pulled like normal cells but are skipped over for level determination. By default, buffers and inverters are considered as one level of logic.

-avoid_soft_blockage

Avoids placing cells in soft placement blockages. By default, hard placement blockage are honored and cells might be placed within soft blockages.

-only_report_magnet_cells

Only report the magnet cells to be pulled while no cells are pulled. This option is mutually exclusive with **-get_collection**.

-get_collection

Return the pulled cells as a collection while pulling them. Use this option with any other options except **-only_report_magnet_cells** to easily get the collection of pulled cells while pulling them within single command.

-stop_points *object_list*

Pulls only the cells that are on the paths between the magnet objects and the given stoppoint objects. The stoppoint objects can be lists of any pin, port, or cell objects.

This option is mutually exclusive with **-logical_levels**, **-exclude_buffers**, **-stop_by_sequential_cells**, and **-cells**.

-multiple_long_port_mode auto | nearby

Specifies how cells are pulled for multiple or long ports. The command supports two modes: auto and nearby.

When 'nearby' set for long port, command pulls placed cells to the nearest location on the port and unplaced cells nearby the port center. When 'nearby' set for multiple port, the command pulls placed cells to one of the nearest port terminals and unplaced cells nearby any one of terminals.

When 'auto' set for a long port, the command places cells evenly near the port. When 'auto' set for a multiple port, command places cells evenly nearby each port terminal.

By default, multiple ports and long port not supported without setting this option.

-hierarchy_mode default | **block** | **all**

Specifies how the command traces cells in the hierarchy in hierarchy placing mode. When this option is specified, all cell instances within each hierarchical block that are connected to the magnet object are pulled toward it. All hierarchies specified by using the **-cells** option are placed one by one.

This option must be used together with the **-cells** option, which by default accepts only flat cell instances as input. When this option is specified, the **-cells** option accepts only hierarchical cell instances as input.

The valid modes are:

- default: The tool traces through only buffers and inverters when traversing from the magnet object to the cells within the specified hierarchical block. This is the default setting.
- block: The tool does not trace through any cell when traversing from the magnet object to the cells within the hierarchical block. The tool pulls the cells of the specified hierarchical block toward the magnet object only if they are directly connected to the magnet object. This is the most restrictive setting.
- all: The tool traces through all combinational cells when traversing from the magnet object to the cells within the specified hierarchical block. However, it does not trace through sequential cells. This is the least restrictive setting.

magnet_objects

Specifies the magnet objects. A magnet object should be a fixed object, for example, a fixed macro cell, a fixed standard cell, a pin of a fixed macro cell, or an I/O pin. Objects like vias and blockages are not allowed to be magnets.

DESCRIPTION

The command defines a set of objects as magnets and pulls the neighboring cells up to a specified logical level closer to the magnet.

It is considered a best practice to run this command before coarse placement, so that sufficient areas are available for magnet placement operation.

A large macro with many placeable standard cell neighbors can be used to ensure that the standard cells are placed close to the macro, resulting in better timing and congestion.

If a net has a fanouts number larger than 1000, the command ignores the net because pulling such a large high-fanout net might cause runtime and congestion issues.

The effect of magnet placement is not preserved from one invocation to another. For example, consider a design where cells A and B are made magnets and there are cells that are connected to both of them through possibly multiple levels. Running **magnet_placement A** and then **magnet_placement B** could result in a different result from running **magnet_placement [get_cells "A B"]** together.

If **magnet_placement** is invoked after **create_placement**, cell overlap issues might occur since there could be standard cells already placed. You can run **legalize_placement** to remove the overlaps.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example performs magnet placement on the *INST_1* cell:

```
prompt> magnet_placement [get_cells "INST_1"]
```

The following example performs magnet placement and marks the moved cells as fixed afterwards:

```
prompt> magnet_placement -mark_fixed [get_cells "INST_2"]
```

The following example pulls two levels of cells to *INST_2*.

```
prompt> magnet_placement -logical_levels 2 \
  [get_cells "INST_2"]
```

The following example pulls the specified cells to INST_3.

```
prompt> magnet_placement INST_3 -cells {Cel_1 Cel_2 Cel_3}
```

SEE ALSO

create_placement(2)
legalize_placement(2)

man

Displays reference manual pages.

SYNTAX

string **man**
topic

Data Types

topic string

ARGUMENTS

topic

Specifies the subject to display. Available topics include commands, variables, and error messages

DESCRIPTION

The **man** command displays the online manual page for a command, variable, or error message. You can write man pages for your own Tcl procedures and access them with the **man** command by setting the **sh_user_man_path** variable to an appropriate value. See the man page for the **sh_user_man_path** variable for details.

EXAMPLES

The following command displays the man page for the **echo** command:

prompt> **man echo**

The following command displays the man page for the error message CMD-025:

prompt> **man CMD-025**

SEE ALSO

help(2)
sh_user_man_path(3)

map_freeze_silicon

Manually maps cells to the specified spare cells.

SYNTAX

status **map_freeze_silicon**
-eco_cell *eco_cell_names*
-spare_cell *spare_cell_names*
[-filler_map_strategy pack_left | pack_right | pack_top | pack_bottom | closest]
| -map_file *map_file_name*
[-lib_cells_for_filler_recovery *psc_filler_lib_cells*]

Data Types

eco_cell_names	list of cells
spare_cell_names	list of cells
map_file_name	string
psc_filler_lib_cells	list of library cells

ARGUMENTS

-eco_cell *eco_cell_names*

Specifies one or multiple cells to be mapped. The specified cells should already exist in the design, whose attribute "is_fs_eco_add" are true and "eco_change_status" are one of create_cell, change_link, add_buffer, add_buffer_on_route, size_cell, eco_placed. If multiple cells are specified and one of them is with error, none of them will be mapped.

This option is mutually exclusive with the **-map_file** option and is used together with the *-spare_cell* option.

-spare_cell *spare_cell_names*

Specifies one or multiple spare cells used for mapping. For programmable spare cell (PSC) mapping, user can specify one or multiple adjacent PSC fillers. But for non-PSC mapping, user can only specify one spare cell. These cells should already exist in the design and they should be spare cells.

This option is mutually exclusive with the **-map_file** option and is used together with the *-eco_cell* option.

-filler_map_strategy pack_left | pack_right | pack_top | pack_bottom | closest

Specifies the location of the ECO cell inside the specified programmable spare cell (PSC) filler. It can only be used when the spare cell is a PSC filler. Default strategy is "pack_left" or "pack_bottom", which means ECO cells will be put one by one from left or bottom of the specified PSC filler. "pack_right" or "pack_top" means ECO cells will be put one by one from right or top of the specified PSC filler. "closest" means a ECO cell will be put at the closest valid location inside the specified PSC filler. "closest" strategy is designed for ECO cells which are already put near expected locations. In this strategy, attribute is_placed of ECO cells must be true.

This option is mutually exclusive with the **-map_file** option and can only be used together with the *-eco_cell* and *-spare_cell* option. When **-map_file** is used, strategy can be specified inside the map file.

-map_file ***map_file_name***

Specifies the name of the map file.

The following is the format of a map file:

```
# Comment should starts by "#". "#" must be the first character of the comment line
# Each line specify a pair of ECO cell and spare cell, separating by a space
eco_cell_1 spare_cell_1
eco_cell_2 spare_cell_2
 ...       ...
# format is same for programmable spare cell (PSC) filler or non-PSC spare
# cell, tool will detect them automatically.
eco_cell_3 psc_filler_3
 ...       ...
# To specify filler_map_strategy, just put it after spare cell (must be a
# PSC filler), separating by a space; If not specified, default is also
# "pack_left" or "pack_bottom"
eco_cell_4 psc_filler_4 pack_top
# To map multi ECO cells to same PSC filler, put them in a {}
# If multiple cells are specified and one of them is with error, none of
# them will be mapped.
{eco_cell_5_1 eco_cell_5_2} psc_filler_5
# It allows that specify multiple PSC fillers in the same mapping pair. But
# these PSC fillers must be adjacent and have the same PSC type. Then they
# can be merged together for mapping large ECO cell or multiple ECO cells.
eco_cell_6 {psc_filler_6_1 psc_filler_6_2}
{eco_cell_7_1 eco_cell_7_2} {psc_filler_7_1 psc_filler_7_2}
```

Error lines will not be mapped, while other lines in the map file still will be mapped. Same ECO cell / PSC filler cannot be specified two times in a map file, otherwise both lines in which they appear will not be executed.

-lib_cells_for_filler_recovery

Specifies the programmable spare cell (PSC) filler lib cells to fill the rest of the specified PSC filler as spare cell, after mapping. When the PSC filler as spare cell is larger than the ECO cell, the rest of it will be filled by newly created PSC fillers of same PSC type. If this option is specified, only lib cells specified will be used for such filling. Otherwise, all available PSC lib cells in the reference libraries will be used.

DESCRIPTION

This command maps the freeze silicon added ECO cell to the specified spare cell, to use the silicon of the spare cell as the ECO cell. The location, orientation, and PG connection of the spare cell will be inherited by the ECO cell. Then the original spare cell itself will be deleted from database. After mapping, an ECO routing command can be used to complete the routing.

In this command, ECO cell must be mapped to spare cell of a compatible lib cell, which means either:

1. ECO cell and spare cell are of same reference lib cell. In this scenario, user can specify only one ECO cell and one spare cell in the mapping pair.

2. Spare cell is programmable spare cell (PSC) filler, whose PSC type is same as the PSC type of the ECO cell or meets PSC mapping rule. In this scenario, user can specify one or multiple PSC fillers with the same PSC type of ECO cells (or meeting PSC

mapping rule) in the mapping pair.
This command will detect these 2 types of mapping automatically.

This command support specify multiple ECO cells or spare PSC fillers. However, this should only be used when it is necessary:

1. Have to specify multiple ECO cells to be mapped by splitting PSC fillers.

2. Have to specify multiple PSC fillers and merge them to map ECO cells. Otherwise, e.g., if you want to map eco_cell1 to psc_filler1 and eco_cell2 to psc_filler2, explicitly specify them in separate **map_freeze_silicon** commands or separate lines in the map file. Otherwise the mapping result might be different than your expectation.

You can specify the ECO cell and spare cell either by using the **-eco_cell** and **-spare_cell** options, or by using a map file.

If PSC filler is larger than the ECO cell, the rest of it will be filled by fillers of same PSC type.

Behavior of this command is same whether the **design.eco_freeze_silicon_mode** application option is true or false.

EXAMPLES

The following example uses **map_freeze_silicon**.

```
prompt> map_freeze_silicon -eco_cell h1_1/U3 -spare_cell spare_buf_0
```

The following example uses **map_freeze_silicon** and specifies a map file.

```
prompt> map_freeze_silicon -map_file map_file
```

The following example uses **map_freeze_silicon** and specifies a programmable spare cell (PSC) filler.

```
prompt> map_freeze_silicon -eco_cell {h1/U3 h2/U4} \
  -spare_cell filler16 -filler_map_strategy pack_right \
  -lib_cells_for_filler_recovery {FILL1 FILL2}
```

SEE ALSO

place_freeze_silicon(2)

map_isolation_cell

Specifies how to map or remap the isolation and enable level-shifter cells belonging to the specified isolation strategy.

SYNTAX

status **map_isolation_cell**
isolation_strategy
-domain *power_domain*
-lib_cells *lib_cells*

Data Types

isolation_strategy	string
power_domain	string
lib_cells	list

ARGUMENTS

isolation_strategy

Specifies the UPF isolation strategy name. The name of the isolation strategy must be unique within the specified power domain.

-domain *power_domain*

Specifies the name of the power domain name to which this isolation strategy belongs.

-lib_cells *lib_cells*

Specifies a list of the target library cells to be used for the isolation mapping.

DESCRIPTION

This command defines how the **compile** command performs the mapping of an isolation cell. All isolation cells that match the strategy will be mapped or remapped to one of the cells specified with the **-lib_cells** option with the highest priority. If no mapped lib cell cannot be used, the tool will select available cell from all libraries. Resizing is done if required; Mapped lib cells have higher priority than library cells.

You can also specify enable level-shifter cells using the **-lib_cell** option. If the isolation cells are later changed to enable level-shifter cells in the technology library, the cells can only be mapped to the one of the enable level-shifter cells specified with this command.

EXAMPLES

In the following example, the isolation cells belonging to the strategy named isolation_1 will be mapped to the LIB_HICLAMP0X2 or LIB_HICLAMP0X4 library cells with the highest priority:

```
prompt> map_isolation_cell isolation_1 -domain PD1 \
 -lib_cells {LIB_HICLAMP0X2 LIB_HICLAMP0X4}
```

SEE ALSO

set_isolation(2)
set_isolation_control(2)

map_level_shifter_cell

Specifies that the level-shifter cells belonging to the specified strategy can only be mapped to a subset of the library cells.

SYNTAX

status **map_level_shifter_cell**
level_shifter_strategy
-domain *power_domain*
-lib_cells *lib_cells*

Data Types

level_shifter_strategy	string
power_domain	string
lib_cells	list

ARGUMENTS

level_shifter_strategy

Specifies the UPF level-shifter strategy name. The level-shifter strategy name must be unique within the specified power domain.

-domain *power_domain*

Specifies the power domain name to which this level-shifter strategy belongs.

-lib_cells *lib_cells*

Specifies a list of library cells that can be used to map to the level-shifter cells.

DESCRIPTION

This command defines how the **compile** command maps the level-shifter cells. All level-shifter cells belonging to the specified strategy are mapped to one of the library cells specified in the **-lib_cells** list with the highest priority. If no library cell is available from the mapped list, the tool will select available cells from all libraries. Resizing is done if required; Mapped lib cells have higher priority than library cells.

If a valid solution cannot be reached using any library cell, no level-shifter cells are inserted. For example, if the level shifters specified in the **-lib_cells** option cannot be used to shift between the required voltage levels, and no other library cell can be used neither, the tool does not insert any level-shifter cell.

For mapping the enable level-shifter cells, use the **map_isolation_cell** command.

EXAMPLES

In the following example the level-shifter cells belonging to the strategy named VL_1 will be mapped to the LIB_LSHIX2 or LIB_LSHIX4 library cells with the highest priority:

```
prompt> map_level_shifter_cell LVL_1 -domain PD1 \
 -lib_cells {LIB_LSHIX2 LIB_LSHIX4}
```

SEE ALSO

map_isolation_cell(2)
set_level_shifter(2)

map_power_switch

Defines which power switch library cells to use for the mapping of the given UPF power switch.

SYNTAX

status **map_power_switch**
switch_name
-domain *domain_name*
-lib_cells *list*

Data Types

switch_name string
domain_name string
list list

ARGUMENTS

switch_name

Specifies the name of the power switch to be mapped using the specified library cells. This is a required option.

-domain *domain_name*

Specifies the power domain where this power switch was created. This is a required option.

-lib_cells *list*

Specifies a list of target library cells to use for the power switch cell mapping. This is a required option.

DESCRIPTION

This command is used to explicitly specify which library power switch cells are mapped for the corresponding UPF power switch.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example maps the switch_1A power switch in the power domain named myPowerDomain with the lib2A/switch2A library cell:

```
prompt> map_power_switch switch_1A \
        -domain myPowerDomain -lib_cells lib2A/switch2A
1
```

SEE ALSO

create_power_switch(2)
get_lib_cells(2)

map_retention_cell

Defines how to map the unmapped sequential cells to retention cells for the specified UPF retention strategy of the power domain.

SYNTAX

status **map_retention_cell**
retention_strategy
-domain *power_domain*
[-lib_cells *lib_cells*]
[-lib_cell_type *lib_cell_type*]
[-elements *objects*]
[-lib_model_name *name* {-port *port_name net_ref*}*]

Data Types

retention_strategy string
power_domain string
lib_cells list
lib_cell_type string
objects list
port_name string
net_ref string

ARGUMENTS

retention_strategy

Specifies the UPF retention strategy name. The retention strategy name should be unique within the specified power domain.

-domain *power_domain*

Specifies the power domain name that this UPF retention strategy will be applied to.

-lib_cells *lib_cells*

Specifies a list of target library lib cells to be used for the retention mapping.

-lib_cell_type *lib_cell_type*

Specifies the retention type to be used for the retention mapping.

-elements *objects*

Specifies that retention mapping only applies to the sequential cells within the specified objects. The objects can be a list of hierarchical cells, leaf cells, or seqgen output signal nets.

-lib_model_name ***name*** **{-port** ***port_name net_ref*****}***

Specifies the retention library cell or behavioral model and associated port connectivity for verification tools.

DESCRIPTION

This command defines how the retention cells are mapped and implemented. In front-end mode, the tool ignores **-elements** and applies this mapping directive to all sequential cells to which the specified retention strategy applies. Also, **-elements** does not get written out in UPF' file.

In back-end mode, the tool ignores **-elements** and applies this mapping directive to all mapped registers to which the specified retention strategy applies. Also, **-elements** does not get written out in UPF' file.

The retention cell specified using the **-lib_model_name** option is not used during implementation. The **-lib_model_name** option is not written out in the UPF' file post compile.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example maps the sequential cells under the *retention_1* retention strategy to the *CLK_LOW* retention type:

```
prompt> map_retention_cell retention_1 \
    -domain PD1 -lib_cell_type CLK_LOW
```

The following example maps the sequential cells under the retention_2 retention strategy to the RSDFCD1HVT and RSDFCD2HVT retention library cells. The **-lib_model_name** option specifies the retention library cell, port name, and net reference used by the verification tools.

```
prompt> map_retention_cell retention_2 \
    -domain PDM -lib_cells { RSDFCD1HVT RSDFCD2HVT } \
    -lib_model_name RSDFCDHVT { -port CP UPF_GENERIC_CLOCK \
                    -port D ~UPF_GENERIC_DATA \
                    -port VDD sn1 \
                    -port TVDD retention_ref.power }
```

SEE ALSO

set_retention(2)
set_retention_control(2)

mark_clock_trees

Marks clock attributes on clock objects.

SYNTAX

status **mark_clock_trees**
[-clocks *clock_list*]
[-clear]
[-synthesized | -dont_touch | -routing_rules]
[-clock_cell_spacing]
[-fix_sinks]
[-freeze_routing]

Data Types

clock_list collection

ARGUMENTS

-clocks *clock_list*

Marks only the clock trees specified in *clock_list*. By default, the command marks for all currently defined clocks in all active modes.

-clear

Removes the attributes on clock objects. If no option other than **-clear** is specified, then tool will remove clock synthesized attributes on clock objects. If the **-dont_touch** option is specified, the tool will remove the dont_touch atrribute on clock objects. Similarly, if any other options are specified together with this option, the tool will clear attributes or settings that the other options are supposed to set on the clock objects.

-synthesized

Marks clock synthesized attributes on clock objects, including cells and nets in the clock trees. On clock nets the net_type attribute updated as clock. The dont_touch attribute for CTS reason is also marked on clock nets. On clock network cells physical_status attribute set as application fixed. On sinks cts_size_in_place attribute set to true. This option cannot be used together with **-dont_touch** and **-routing_rules**.

-dont_touch

Sets dont_touch attribute on clock objects, including cells and nets in the clock trees. This option cannot be used together with **-synthesized** and **-routing_rules**.

-routing_rules

Propagates the non-default routing rules according to the settings of clock routing rules by **set_clock_routing_rules** command.

This option cannot be used together with **-synthesized** and **-dont_touch**.

-clock_cell_spacing

Propagates the clock cell spacing to the clock cells according to the rules specified by **set_clock_cell_spacing** command.

-fix_sinks

Marks clock sink as fixed placement in order to avoid subsequent commands sizing or moving them.

-freeze_routing

Marks synthesized net as locked and dont_touch in order to avoid routing of net by subsequent commands. This option cannot be used together with **-routing_rules**.

DESCRIPTION

This command marks clock attributes on clock objects to prevent the subsequent optimization programs to modify the built clock trees. Since the **synthesize_clock_trees** command automatically marks attributes on clock objects, the motivation of this command is for the third party tools. Default behavior of mark_clock_trees is "synthesized" mode.

EXAMPLES

The following example assumes a clock tree clk1 synthesized by a third party tool. After **mark_clock_trees**, this clock tree can then be recognized by the optimization programs.

```
prompt> mark_clock_trees
```

In the following example, default behavior of clear option is shown. It will only clear settings done by "synthesized" mode.

```
prompt> mark_clock_trees -clear
```

SEE ALSO

synthesize_clock_trees(2)
set_clock_cell_spacing(2)

mem

Retrieves the total memory allocated by the current process.

SYNTAX

int **get_mem**

DESCRIPTION

The *get_mem* command returns the size of memory currently allocated by the current process for the purposes of storing design netlist, design annotations, and constraints information as well as computed timing information. The value printed represents the amount in kilobytes (kB).

Note that the value reported would not match values obtained by the user through Unix process tracking commands, such as *top*. This is the case because the *get_mem* command does not track memory used to load the executable and maintain process stack space. Also, it does not differentiate between resident and swapped memory.

```
prompt> get_mem
8092
```

SEE ALSO

get_cputime(2)

merge_abstract

Merges changes made to the current design's abstract view into the full design view.

SYNTAX

status **merge_abstract**
[-blocks *block_designs*]
[-all_blocks]
[-host_options *host_option_name*]
[-force]

Data Types

block_designs list *host_option_name* string

ARGUMENTS

-blocks *block_designs*

Specifies the list of child block designs on which to perform merge. This option is mutually exclusive with *-all_blocks*. If the *-blocks* or *-all_blocks* options are not specified, the command merges the current design.

-all_blocks

Merges abstracts for all the child block instances in the current design. This option is mutually exclusive with *-blocks*. If the *-blocks* or *-all_blocks* options are not specified, the command merges the current design.

-host_options *host_option_name*

Specifies the name of the host option to use for distributed processing. When *-blocks* or *-all_blocks* is specified, the tool might need to perform merge for more than one block. Some of the blocks can run at the same time, depending on the hierarchy nesting.

-force

Merges changes to the abstract view for the current design into the full design view, regardless of the current state of the views.

DESCRIPTION

This command updates the full netlist design view for the current design with any changes made to the abstract view.

Use the **merge_abstract** command to keep the design view representation up-to-date. After running **merge_abstract**, you can

immediately run the **create_abstract** command to generate a new abstract view from the up-to-date design view, or you can perform other operations before creating a new abstract view.

This command merges any changes to the abstract view into the design view. Possible changes include: netlist modification, boundary shape change, cell location change, newly created or deleted objects, objects pushed down from higher level of physical hierarchy, and so on.

By default, abstracts created with **create_abstract** can be edited. That is, you can open the abstract in edit mode and make changes. Changes can be merged back into the full netlist design by using the **merge_abstract** command.

Design Flow for Editable Abstracts

When you create an editable abstract (the default), it is assumed that you will eventually run **merge_abstract** after editing the abstract. The **merge_abstract** command requires that the full netlist design remain unchanged between the time that the abstract is created with the **create_abstract** command and the abstract is merged with the **merge_abstract** command.

Because the netlist should not be changed, the **create_abstract** command marks the full netlist as read-only until after running the **merge_abstract** command. After you run **merge_abstract**, the open blocks that depend on the abstract are automatically upgraded to "edit" mode.

EXAMPLES

The following example opens the design view of CHIPTOP and merges the changes to the abstract view back to the design view of design CHIPTOP.

```
prompt> open_block CHIPTOP.design
prompt> merge_abstract
```

SEE ALSO

change_abstract(2)
create_abstract(2)
get_abstract_type(2)
remove_abstract(2)
report_abstracts(2)

merge_clock_gates

Merges ICG cells in clock trees.

SYNTAX

status **merge_clock_gates**
[-clocks *clock_list*]

Data Types

clock_list list

ARGUMENTS

-clocks *clock_list*

Merges ICG cells only in the clock trees specified in *clock_list*. By default, the command merges ICG cells for all currently defined clocks in all active modes.

DESCRIPTION

This command merges ICG cells in each clock tree without affecting the logic of the design. The merged ICG cells have the same type and their enable pins have the same logic.

EXAMPLES

The following example merges ICG cells in the specified clock tree.

prompt> **merge_clock_gates -clocks clk1**

SEE ALSO

merge_objects

Merges shapes of the specified objects.

SYNTAX

status **merge_objects**
[*object_list*]
[-attributes *attribute_list*]
[-connections]

Data Types

object_list string or collection
attribute_list list of attribute names

ARGUMENTS

object_list

Specifies a collection or selection set containing objects to merge. If this option is not specified, the global selection set is used.

-attributes *attribute_list*

Specifies additional list of attributes which need to match in order to merge two shapes. By default merge happens when **object_class**, **layer**, **shape_use** and **owner** attributes match.

Note: The additional attribute are being compared by the string values.

-connections

If specified collection or selection set contains shapes and vias, lookup for the first level connections to merge with.

DESCRIPTION

This command merges specified object shapes; all fixed objects are skipped. The following objects are supported: move bound shapes, voltage area shapes, corridor shapes, net shapes, and pin shapes.

Snapping is not performed as it is assumed that the shapes are already correct.

Note that only interconnected net shapes that belong to the same net, are on the same metal layer, and have the same width and end type can be merged. Nonrectilinear ending segments of a path are not merged.

The command returns a collection of the merged objects if the specified *object_list* is a collection of objects. Otherwise the command modifies objects in specified selection set and returns status.

EXAMPLES

The example merges two shapes to a single shape.

```
prompt> merge_objects
```

The alternative syntax uses collection to specify objects.

```
prompt> merge_objects [get_selection]
```

SEE ALSO

change_selection(2)
create_shape(2)
get_selection(2)
get_shapes(2)
split_objects(2)

merge_pg_mesh

Merges overlapped wires of PG nets after the PG mesh is created.

SYNTAX

status **merge_pg_mesh**
[-nets *netname_list*]
[-types *type_list*]
[-layers *layer_list*]
[-shapes *shape_list*]
[-undo]

Data Types

netname_list list
type_list list
layer_list list
shape_list list

ARGUMENTS

-nets *net_list*

Specifies a list of PG nets on which to merge wires. By default, all PG nets are processed.

-types *type_list*

Specifies a list of PG object types. The PG types should include one or more of the following keywords: **stripe**, **ring**, **macro_pin_connect**, **lib_cell_pin_connect**, **follow_pin**, **core_wire**, and **user_route**. These types are for PG stripes, PG rings, macro pin connections, standard cell pin connections, follow pins wires, core wires and user route wires. By default, all of these types are considered for wire merging.

-layers *layer_list*

Specifies metal layer names for the PG objects. *layer_list* is a list containing layer names. By default, objects on all metal layers are considered for wire merging.

-shapes *shape_list*

Specifies a list of PG shapes for which to merge wires. By default, only top-level wires of PG nets will be merged.

-undo

Restores merged shapes to their original objects that merged by most recent **merge_pg_mesh** command. Only one level of undo is supported. The **merge_pg_mesh -undo** command clears the undo command stack.

DESCRIPTION

This command performs PG wire merging after the PG mesh is created for the whole design. The command is use to reduce PG wire objects without changing wire shape geometry. The command only merge rectangle shapes. If the two overlapped shapes cannot form a new rectangle, they won't be merged. The **merge_pg_mesh** command only merges wire shapes on top-level. After **merge_pg_mesh**, one level undo is supported for user to recover original shapes.

EXAMPLES

The following example performs PG wire merging on all PG nets. after PG mesh is created.

```
prompt> merge_pg_mesh
```

The following example performs PG wire merging with some constraints. Only wire shapes with those attribute are process.

```
prompt> merge_pg_mesh \
   -nets {VDD VSS} \
   -types {stripe lib_cell_pin_connect} \
   -layers M5
```

SEE ALSO

compile_pg(2)
create_pg_composite_pattern(2)
create_pg_macro_conn_pattern(2)
create_pg_ring_pattern(2)
create_pg_std_cell_conn_pattern(2)
create_pg_strap(2)
get_shapes(2)
set_pg_strategy(2)

merge_stream

Merges multiple GDSII or OASIS stream files to a single stream file.

SYNTAX

status **merge_stream**
[-format *output_format*]
[-rename_cell *cell_renaming_file*]
[-merge_cell_shapes]
[-rename_conflicting_cell]
[-merge_conflict_suffix *suffix_string*]
[-compress_gds]
[-oasis_compression_level *level*]
[-cell_log_file *cell_log_file*]
[-units *value*]
[-top_cell *cell_name*]
[-verbose]
-merge_files *file_names*
output_file

Data Types

output_format string
cell_renaming_file string
suffix_string string
level integer
cell_log_file string
value double
cell_name string
file_names list of strings
output_file string

ARGUMENTS

-format *output_format*

Specifies the output file format, either **gds** for GDSII or **oasis** for OASIS. The default is **gds**.

-rename_cell *cell_renaming_file*

Specifies the file that contains the mapping from original cell names to new cell names. For the renaming file format, see the DESCRIPTION section. The renaming works for both leaf-level library cells and macros. By default, the merging process keeps the original cell names.

-merge_cell_shapes

Specifies that the shapes from identically named cells in different source files are to be merged into a single cell in the output file. The first source file must contain all the cells' structure definitions. If one cell's structure definition is not included in the first source file, it will not be merged into the output file. By default, while reading the source files, the command overwrites any previous cell of the same name, without performing any merging or renaming of the conflicting cells.

-rename_conflicting_cell

Specifies that identically named cells in different source files are to be renamed to avoid conflict. The first cell read in keeps its original name. Any subsequent cells with the same name are renamed by appending a string and then written to the output file as separate cells. The default strings appended are _R, _R1, _R2, and so on. By default, while reading the source files, the command overwrites any previous cell of the same name, without performing any merging or renaming of the conflicting cells.

-merge_conflict_suffix *suffix_string*

Specifies the suffix string used with the **-rename_conflicting_cell** option. The string can be up to 10 characters and may contain letters, numerals, dollar signs ($), and underscore characters. The default suffix is "_R".

-compress_gds

Writes out the GDSII file in gzip compressed format. By default, the output GDSII file is written as uncompressed plain text.

-oasis_compression_level *level*

Writes out the OASIS file in compressed format using the specified compression level, an integer between 1 and 9. By default, the output OASIS file is not compressed. This option has an effect only if the **-format** option is set to **oasis**.

-cell_log_file *cell_log_file*

Specifies the name of a cell log file. The command writes out the names of the processed cells into this file.

-units *value*

Specifies the database unit size for the output stream file. The default is 1e-09 (nanometer units).

-top_cell *cell_name*

The top cell considered in the merge flow. The command merges only the specified cell and the cells in its hierarchy. All other cells are omitted from the output file. By default, the command writes out all cells in the input files.

-verbose

Generates detailed system messages during stream merging.

-merge_files *file_names*

Specifies the names of the input stream files to be merged. You can specify both GDSII and OASIS files as input.

output_file

Specifies the name of the output file generated by merging the input files.

DESCRIPTION

This command merges multiple GDSII/OASIS stream files into a single stream file in GDSII or OASIS format. You must specify at least the name of the stream files to be merged and the name of the new file generated by the merging process.

You can optionally specify the handling of identically named cells in different input files. By default, the last cell read in overwrites any earlier cells of the same name. Instead, you can merge the data in these cells (**-merge_cell_shapes**), or you can rename the

cells to avoid conflict.

To rename cells, you can have a string automatically appended to each cell name (**-rename_conflicting_cell**). Alternatively, you can specify the new cell names explicitly in a file (**-rename_cell** *filename*), with each line in the file specifying the name mapping for a cell in the following format:

OldCellName NewCellName

For example, to rename the cell TOP to NEWTOP, enter the following line in the renaming file:

TOP NEWTOP

EXAMPLES

The following example merges the colored macro named top.gds with the IP hard macros named HM1.gds and HM2.oasis:

```
prompt> merge_stream -format gds -units 1e-09 \
  -merge_files {top.gds HM1.gds HM2.oasis} my_merge.gds
```

SEE ALSO

write_gds(2)
write_oasis(2)

merge_topology_plans

SYNTAX

merge_topology_plans
[-repeaters]
[-supernets]
[-verbose]

[*topology plans*]

Data Types

topology plans collection or list of names of topology plans to estimate

merge_topology_plans.

ARGUMENTS

-repeaters

Only attempt to associate std cell register arrays with topology repeaters.

If neither *-repeaters* or *-supernets* is specified, both actions are attempted.

-supernets

Only attempt to create new supernet exceptions for topology plans with std cell repeaters in the netlist for which supernet exceptions have not been created. Do not attempt to associate repeaters.

If neither *-repeaters* or *-supernets* is specified, both actions are attempted.

-verbose

Print additional information at the end of processing each topology plan as to actions taken for that topology plan.

DESCRIPTION

This command can be used to process topology plans after various actions that would invalidate them for further processing. For example, adding new std cell registers to the netlist that is associated to a topology plan, without adding new supernet exceptions, or associating those registers to topology repeaters.

The supernet exception creation operation scans the netlist from the terminals associated with the end(s) of a topology plan to the terminals associated with the start of that plan and records any register cell encountered. If any such cells do not have a corresponding supernet exception, one is created for that cell.

The repeater association operation scans the netlist from the terminals associated with the end(s) of a topology plan to the terminals associated with the start of that plan and records any register cell encountered. If any such cells are not associated with a topology repeater, and a valid un-associated topology repeater exists in the plan, the std cells are associated with the topology repeater.

By default (no arguments) both operations are performed: supernet exception creation, followed by repeater association. All specified topology plans are first processed for supernet creation, and then all specified topology plans are processed for repeater association.

KNOWN LIMITATIONS

Currently, new topology repeaters are not created, only existing topology repeaters are assigned std cells.

SEE ALSO

create_topology_plan(2)
create_topology_repeater(2)
get_topology_plans(2)
report_topology_plans(2)
report_topology_repeaters(2)
report_supernet_exceptions(2)
set_supernet_excpetions(2)

modify_busplan

Modify various elements of a busplan.

SYNTAX

int **modify_busplan**
[-add_guide]
[-add_register]
[-move_element *element*]
[-balance_registers]
[-remove_elements *elements*]
[-reconnect_elements *elements*]
[-change_rule *rule*]
[-after *element*]
[-before *element*]
[-location *coord*]
[-name *element_name*]
[-rename *new_name*]
[-load_launch_capture_budget]
[*busplan*]

Data Types

element	bus_element
elements	bus_elements
rule	bus_rule
coord	coordinate
element_name	string
new_name	string
busplan	busplan_name

ARGUMENTS

-add_guide

Adds a guide element to the bus plan.

-add_register

Adds a virtual register to the bus plan.

-move_element *element*

Changes the location of *element*. You must specify the *-location* option together with this option.

-balance_registers

Adds a minimal number of registers such that each branch of the busplan has the same number of registers. This only applies to busplans that have branches.

-remove_elements *elements*

Removes the specified elements from the busplan. It is an error to try to remove a start or end element of a busplan.

-reconnect_elements *elements*

Removes and reconnects the path tree starting from *elements* to a new position in busplan. This is likely to change the topology of the bus plan. You must specify the *-after* option together with this option.

-change_rule *rule*

Specifies the net estimation rule to use.

-after *element*

Specifies the position in the bus to connect a newly created element with the *-add_guide* or *-add_register* options, or reconnect an existing element with the *-reconnect_elements* option.

-before *element*

Specifies the position in the bus to connect a newly created element with the *-add_guide* or *-add_register* option.

-location *coord*

Specifies the physical location for a newly created element with the *-add_guide* or *-add_register* options, or specifies a location for *-move_element*. This option is required when you specify *-move_element*.

-name *element_name*

Specifies the name for the newly created element. If not specified, a default name is created.

-rename *new_name*

Specifies a new name for the busplan.

-load_launch_capture_budget

Specifies to load the launch and/or capture budget from the current SDC constraints. This only works for top level ports on the busplan. The launch budget is gotten from the **set_input_delay** values on the starting elements of the busplan. The capture budget is gotten from the **set_output_delay** values on the ending elements of the busplan.

busplan

Specifies the busplan to be modified. This is required.

DESCRIPTION

This command changes the specified busplan. These changes include structural changes as well as placement changes.

You must specify one of the following options to indicate the change required: *-add_guide*, *-add_register*, *-move_element*, *-remove_elements*, *-reconnect_elements*, or *-change_rule*.

Depending on the change required, other options then are required. See above for more details.

EXAMPLES

The following example creates a bus plan, then adds a virtual route pin to the bus plan.

```
prompt> create_busplans -name bus1 -from B1/OUT1[*] -start_end_cells {B1 B2 B3 B4}
prompt> report_busplans bus1

****************************************
Report : report_busplans
Design : TOP
Version: L-2016.03-DEV
Date   : Sun Aug  2 16:55:49 2015
****************************************
Start/end at instances: B1 B2 B3 B4
bus1: 8 bits start=B1/OUT1[7] clock=-unknown-
  pin               group1     (  8) B1/OUT1[7]
    register          group2     (  8) r1/r7
      register        group3     (  8) r2/r7
        pin           group4     (  8) B2/IN1[7]
1

prompt> modify_busplan -add_guide -after group1 -name aGuide bus1
{aGuide}

prompt> report_busplans bus1
****************************************
Report : report_busplans
Design : TOP
Version: L-2016.03-DEV
Date   : Sun Aug  2 16:55:49 2015
****************************************
Start/end at instances: B1 B2 B3 B4
bus1: 8 bits start=B1/OUT1[7] clock=-unknown-
  pin               group1     (  8) B1/OUT1[7]
    guide             aGuide
      register        group2     (  8) r1/r7
        register      group3     (  8) r2/r7
          pin         group4     (  8) B2/IN1[7]
1
```

SEE ALSO

create_busplans(2)
get_busplans(2)
report_busplans(2)
report_net_estimation_rules(2)
set_net_estimation_rule(2)
write_busplans(2)

modify_rp_groups

Modifies relative placement group by adding, removing, swapping or flipping rows or columns.

SYNTAX

```
status modify_rp_groups
  rp_group_list
  -add_rows          row_number    number_of_rows    |
  -add_columns       column_number number_of_columns |
  -remove_rows       row_number    number_of_rows    |
  -remove_columns    column_number number_of_columns |
  -flip_row          row                             |
  -flip_column       column                          |
  -swap_rows         first_row     second_row        |
  -swap_columns      first_column  second_column
```

Data Types

```
rp_group_list      list or collection
row_number         integer
column_number      integer
number_of_columns  integer
number_of_rows     integer
row                integer
column             integer
first_row          integer
second_row         integer
first_column       integer
second_column      integer
```

ARGUMENTS

rp_groups_list

Specifies the relative placement groups that should be modified.

-add_rows *row_number number_of_rows*

Specifies two arguments to add new rows to relative placement group. The first argument is the row number where you want to add new rows and the second argument specifies the number of consecutive rows to be added.

-add_columns *column_number number_of_columns*

Specifies two arguments to add new columns to relative placement group. The first argument is the column number where you want to add new columns and the second argument specifies the number of consecutive columns to be added.

-remove_rows ***row_number number_of_rows***

Specifies two arguments to remove rows from relative placement group. The first argument is the row number where you want to remove rows and the second argument specifies the number of consecutive rows to be removed.

-remove_columns ***column_number number_of_columns***

Specifies two arguments to remove columns from relative placement group. The first argument is the column number where you want to remove columns and the second argument specifies the number of consecutive columns to be removed.

-flip_row ***row***

Specifies the row to be flipped. This option flips the objects in the row around the center of the row.

-flip_column ***column***

Specifies the column to be flipped. This option flips the objects in the column around the center of the column.

-swap_rows ***first_row second_row***

Specifies two rows of the specified relative placement group to be swapped. This option swaps the objects (cells, blockages, and hierarchical relative placement groups) of two specified rows.

-swap_columns ***first_column second_column***

Specifies two columns of a relative placement group to be swapped. This option swaps the objects (cells, blockages and hierarchical relative placement groups) of two specified columns.

DESCRIPTION

This command modifies the structure of a specified relative placement group. You can add, remove, swap, or flip rows or columns of the specified relative placement group. If after modification, the location for *rp_location* anchor corner is empty then anchor corner will be removed from relative placement group.

EXAMPLES

The following example adds 3 new columns to all existing top-level relative placement groups at column position 0.

```
prompt> modify_rp_groups  [get_rp_groups -top] -add_columns {0 3}

1
```

The following example flips the first row of all top-level relative placement groups.

```
prompt> modify_rp_groups [get_rp_groups -top] -flip_row 0
1
```

The following example swaps the first and second columns of the my_rp_group relative placement group:

```
prompt> modify_rp_groups [get_rp_groups my_rp_group] -swap_columns {0 1}
1
```

The following example removes 2 rows from all existing top-level relative placement groups at row position 1

```
prompt> modify_rp_groups  [get_rp_groups -top] -remove_rows {1 2}

1
```

SEE ALSO

create_rp_group(2)
get_rp_groups(2)
write_rp_groups(2)

move_block

Moves a block to a new block. The block can be moved to a different library or view.

SYNTAX

block_return **move_block**
[-force]
[-from_block *blocks*]
[-lib *lib*]
-to_block *block_name*

Data Types

block_return Collection containing block or "".
block Block name in [libName:]blockName[/labelName][.viewName] format or a collection of one block.
block_name Name in form [libName:]designName[/labelName][.viewName]

ARGUMENTS

-force

Moves the block to the destination, even if it is opened and unsaved.

-from_block *blocks*

Specifies the source block name with optional library, label, and view specifications or a collection of blocks. If the library specification is not present, the **current_lib** is used. If the label specification is not present, then the default unnamed label is used. If the view specification is not present, the *design* view is used. By default, the **current_block** is used if this option is not specified.

-to_block *block_name*

Specifies the destination block name with optional library and view specifications. If the library, block, or view specifications are not present, the values from the source block are used.

-lib *lib*

Specifies the library to which the collection of blocks to be moved to. This option is exclusive to option **-to_block**.

DESCRIPTION

This command moves a source block(s) to the specified destination with the same or different library and block. Note that it is an error to attempt to move a block over itself. The command fails if the following conditions exist:

- The source library is opened in read-only mode
- The destination library is not open or is opened in read-only mode
- The destination block is open and has been modified but not saved and the **-force** option is not used
- An illegal name is given for the source or destination library, block, or view.
- The source block is a library cell block timing view. These views must be built "in place" in the library Manager because of library-level data.
- The destination block view is different than the source block view.

If successful, the destination block(s) can be further modified without any change in open mode. Note that the operation of this command is identical to the **copy_block**, except that the source block is removed (from memory, from the library, and from disk) after the block is successfully moved into its new location.

The command returns the block(s) if successful, or O if not. If an illegal name is given for the source or destination block names, a Tcl error is raised.

When no view is explicitly specified for both source and destination then all available views of source block are moved to destination block and all the moved blocks are returned.

EXAMPLES

This example moves a block *Top* to *Top2* to rename it. The destination library and view are taken from the original block.

```
prompt> move_block -from_block Top -to_block Top2
{"Lib:Top2.design"}
```

This example moves a block Mid into a new library RefLib1.

```
prompt> move_block -from_block Mid -to_block RefLib1:
{"RefLib1:Mid.design"}
```

SEE ALSO

close_blocks(2)
copy_block(2)
create_block(2)
get_blocks(2)
get_designs(2)
open_block(2)
open_lib(2)
remove_blocks(2)
reopen_block(2)
save_block(2)

move_block_origin

Moves a block's origin to a new point in its own coordinate space.

SYNTAX

status **move_block_origin**
[-to *point*]
block

Data Types

point list of two floats: *x* and *y*
block Block name in [libName:]blockName[/labelName][.viewName] format or a collection of one block.

ARGUMENTS

-to

The point in *block*'s own coordinate space to become its new origin.

block

The block to move the origin of. If unspecified, the current block's origin will be moved.

DESCRIPTION

This command moves a block's origin to a new location in its own coordinate space. The new origin becomes (0, 0), and block's contents are shifted accordingly.

EXAMPLES

The following example moves the current block's origin to (100, 100), and illustrates the change in coordinates of one of block's member cells:

```
prompt> get_attribute [get_cells bot_0] origin
0.1250 -12.6000
```

```
prompt> get_attribute [get_cells bot_0] bbox
{0.1050 -12.6200} {12.6250 -0.1000}

prompt> move_block_origin -to {100 100}

prompt> get_attribute [get_cells bot_0] origin
-99.8750 -112.6000

prompt> get_attribute [get_cells bot_0] bbox
{-99.8950 -112.6200} {-87.3750 -100.1000}
```

SEE ALSO

set_attribute(2)
get_attribute(2)
get_blocks(2)
current_block(2)
move_objects(2)

move_lib

Moves a library and its contents from one place to another.

SYNTAX

lib_return **move_lib**
[-force]
[-from_lib *library*]
-to_lib *lib_name*

Data Types

lib_return Collection containing target library or "".
library Name or collection containing the source library.
lib_name name of the library.

ARGUMENTS

-force

This option is used to overwrite the destination library when the library is open, modified, but not yet saved.

-from_lib *lib_name*

Source library. If not given, the current_lib is used. This can either be a (full-path, relative-path, or simple) name, or a collection of a single library.

-to_lib *lib_name*

Specifies the destination library name.. This can be a simple, relative, or absolute path. If it is a relative or absolute path, the trailing portion of the path after he last '/' becomes the name of the library. It will be an error if the a library, file, or directory of the same name already exists on disk.

RETURN VALUE

Returns the library if successful, or 0 if not. If an illegal name is given, a TCL error is raised.

DESCRIPTION

This command moves a library from one place to another in the file system. It is largely unnecessary as the LINUX mv(1) command will work also, but it is in our system form completeness. For both **copy_lib** and **move_lib**, care is taken to not read into memory any of the designs that are already in memory, so copying a large library is as efficient as it can be.

EXAMPLES

This example moves RefLib1, calling it RefLibA.

```
prompt> move_lib -from_lib RefLib1 -to_lib RefLibA
{"RefLibA"}
```

SEE ALSO

create_lib(2)
open_lib(2)
save_lib(2)
close_lib(2)
copy_lib(2)
current_lib(2)
get_libs(2)
set_ref_libs(2)
search_path(3)

move_objects

Moves and rotates the specified objects.

SYNTAX

status **move_objects**
[-delta *delta*]
[-from *point*]
[-to *point*]
[-x *x*]
[-y *y*]
[-rotate_by *angle*]
[-group]
[-force]
[-simple]
[*object_list*]

Data Types

object_list collection or selection
delta list of two floats: *x* and *y*
point list of two floats: *x* and *y*
x float
y float
angle float

ARGUMENTS

object_list

Specifies a collection or selection set containing objects to move. If this option is not specified, the global selection set is used.

-delta *point*

Specifies the distance in the x- and y-directions to move the objects from the object's current location. This option is mutually exclusive with the **-to**, **-from**, **-x** and **-y** options.

-from *point*

Specifies the reference point on the object, or the collection of objects to be moved, for the **-to** option. By default, the command uses the lower-left corner of the bounding box of the given object, or collection of objects, as the reference point. This option must be used together with the **-to** option.

-to *point*

Specifies the new location of the reference point for the object or objects. This option is mutually exclusive with the **-delta**, **-x** and **-y** options.

-x ***float***

Specifies the x-coordinate for the given object or collection of objects. All objects are moved so that their left edge is placed at the specified x-coordinate. This option is mutually exclusive with the **-delta**, **-from** and **-to** options.

-y ***float***

Specifies the y-coordinate for the given object or collection of objects. All objects are moved so that their bottom edge is placed at specified y-coordinate. This option is mutually exclusive with the **-delta**, **-from** and **-to** options.

-rotate_by ***angle***

Specifies the rotation angle of the objects.

The valid values are:

0, **90**, **180**, **270**,
CW90, **CW180**, **CW270**, **CCW90**, **CCW180**, **CCW270**
FLIPX, **FLIPY**
R0, **R90**, **R180**, **R270**, **MX**, **MXR90**, **MY**, **MYR90**

The rotation value definitions are:

CW90 is 90 degrees clockwise
CW180 and **CCW180** are 180 degrees
CW270 is 270 degrees clockwise
CCW90 is 90 degrees counterclockwise
CCW270 is 270 degrees counterclockwise
FLIPX is the reflection about the x-axis
FLIPY is the reflection about the y-axis

The following values are synonymous:

0, **R0**
90, **CCW90**, **R90**
180, **CCW180**, **R180**
270, **CCW270**, **R270**,
FLIPX, **MY**
FLIPY, **MX**

-group

Specifies if rotation should be made around common pivot point. If omitted, the rotation is performed around the object centers.

-force

Resizes locked or fixed objects. By default, such objects are not resized according to global edit settings. Edit settings can be accessed using **get_edit_setting**.

-simple

Disable snapping and editing constraints. By default, objects are snapped according to global snap settings. Snap settings can be accessed using **get_snap_setting**.

DESCRIPTION

This command moves one or more specified objects to a given location skipping all fixed or locked objects. The command keeps pin/terminal on edge of parent block and updates its layer if corresponding edit setting is set. See the **set_edit_setting** command for details.

Snapping is done automatically using the global snap settings.

EXAMPLES

The following example moves all selected objects by delta (-100 100).

prompt> **move_objects -to {-100 100} -from {0 0}**

The alternative syntax uses collection to specify objects.

prompt> **move_objects [get_selection] -to {-100 100} -from {0 0}**

SEE ALSO

change_selection(2)
copy_objects(2)
get_edit_setting(2)
get_selection(2)
get_snap_setting(2)
rotate_objects(2)
set_edit_setting(2)
set_snap_setting(2)
snap_objects(2)

name_format

Specifies the name format for the isolation cells and level shifters.

SYNTAX

status **name_format**
[-isolation_prefix *name*]
[-isolation_suffix *name*]
[-level_shift_prefix *name*]
[-level_shift_suffix *name*]

Data Types

name string

ARGUMENTS

-isolation_prefix

Specifies the prefix to use for the isolation cell names. The default is "".

-isolation_suffix

Specifies the suffix to use for the isolation cell names. The default is "_UPF_ISO".

-level_shift_prefix

Specifies the prefix to use for the level-shifter cell names. The default is "".

-level_shift_suffix

Specifies the suffix to use for the level-shifter cell names. The default is "_UPF_LS".

DESCRIPTION

This command specifies the suffix and prefix for the isolation and level-shifter cells. The name of the element that is being isolated or level shifted by the isolation or the level-shifter cell is used in the middle. For example, an isolation cell that isolates a port named *p1* is named *p1_UPF_ISO* by default.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of the **name_format** command:

```
prompt> name_format -isolation_prefix "MY_ISO" -level_shift_prefix "MY_LS"
```

SEE ALSO

set_isolation(2)
set_level_shifter(2)

open_attachment

Opens a file handle for the given attachment.

SYNTAX

status **open_attachment**
names
-mode *file_mode*
[-of_object *object*]

Data Types

names list
file_mode string
object collection

ARGUMENTS

names

List of attachment names to be reported. Supports glob style patterns.

-mode *file_mode*

Specifies the mode for opening the file. All the modes that work with Tcl "open" command are supported.

-of_object *object*

Specifies the object whose attachments will be opened. By default the current block and current library will be considered in that order as the owner object.

DESCRIPTION

This command opens a file handle for the attachment with the given name and for the specified owner object. The file handle is open in the mode specified. The command should be used in a similar fashion as the Tcl "open" command to open the file handle.

EXAMPLES

In the following example the file handle is opened and used for the given attachment.

```
prompt> set fh [open_attachment -of_object [current_block] -mode r test_att]
 file23
prompt> read $fh
 ATTACHMENT CONTENTS
```

SEE ALSO

add_attachment(2)
remove_attachments(2)
report_attachments(2)

open_block

Opens an existing saved block for editing or viewing and sets that block as the current block.

SYNTAX

collection **open_block**
[-edit | -read | -check]
[-ref_libs_for_edit]
[*block_name*]

Data Types

block_name string

ARGUMENTS

-edit

Opens the block in edit (read/write) mode, allowing you to modify and save the block. This is the default mode for opening a block in a design library. To open a block in edit mode, you must have write permission for the directory containing the block and library, and the block must not be already locked for editing by another user.

-read

Opens the block in read-only mode, allowing you to view but not modify or save the block.

-check

Checks for existence of the block in memory. Returns the block if it is already in memory, or returns an error message otherwise. Mutually exclusive with **-read** and **-edit**.

-ref_libs_for_edit

Opens lower-level design libraries (those appearing in the reference library list) in edit mode so that lower-level blocks can be modified. Reference libraries containing lib_cell blocks remain in read-only mode, even when this option is used. This option can only be used when opening a block in edit mode, either by using the default open mode or explicitly with **-edit** option.

block

A collection of one block or name of the block to open, with optional *library*, *label*, and *view* names specified. The default names are the current library, no label, and the design view.

DESCRIPTION

This command opens an existing saved block for editing or viewing and sets that block as the current block. The block is opened in edit mode by default or when the **-edit** option is used. The block is opened in read-only mode when the **-read** option is used. With the **-check** option, the command returns the block as a collection if it is already in memory, or an error message otherwise.

By default, when you open a block in edit mode, when the block is in an open read-only library, the command promotes the library to edit mode so that you can edit the block in the library. To prevent this action and protect the library from being accidentally modified, set the **design.edit_read_only_libs** application option to false.

An implementation tool can open library cell blocks in cell libraries only in read-only mode. To edit library cell blocks, use the library manager tool (lm_shell).

The **open_block** command increments the "open count" for a block (unless you are using the **-check** option). The open count can affect the read/write status of the block. For example, if you open a block using **-edit** and the again using **-read**, the read/write status of the block remains in "edit" mode due to the status of the earlier open operation (earlier open count).

EXAMPLES

The following example opens the existing block "Top" from the library "MyDesigns":

```
prompt> open_block -edit MyDesigns:Top
{"MyDesigns:Top.design"}
```

The following example opens the timing view of lib_cell AN2 from the library MyLibCells:

```
prompt> open_block -read MyLibCells:AN2.timing
{"MyLibCells:AN2.timing"}
```

SEE ALSO

create_block(2)
close_blocks(2)
copy_block(2)
move_block(2)
remove_blocks(2)
reopen_block(2)
save_block(2)
open_lib(2)
get_blocks(2)
current_block(2)
get_designs(2)
current_design(2)
design.edit_read_only_libs(3)

open_drc_error_data

Opens an error data file so that you can query or modify the error data.

SYNTAX

collection **open_drc_error_data**
[*drc_error_data*]
[-file_name *file_name*]
[-readwrite]
[-readonly]
[-checkonly]

Data Types

drc_error_data collection
file_name string

ARGUMENTS

drc_error_data

Specifies the attached error data file to open. You can use the file name, such as zroute.err, or a collection that contains the error data file.

To determine the error data files attached to the current block, use the **get_drc_error_data -all** command. You must use the **-all** option to return both the open and unopened error data files.

This argument is mutually exclusive with the **-file_name** option; you must specify one of these arguments.

-file_name *file_name*

Specifies the file path of an external error data file to open.

This argument is mutually exclusive with the *drc_error_data* argument; you must specify one of these arguments.

-readwrite

Opens the specified external error data file in read/write mode.

When you enable this mode, the tool can modify the contents of the error data file. If you open a file in read/write mode, it stays in that mode even if you later open it in read-only mode.

This is the default open mode for external error data files, which are opened with the **-file_name** option.

The tool ignores this option when opening attached error data files and uses the open mode of the block.

The **-readwrite**, **-readonly**, and **-checkonly** options are mutually exclusive; you can specify only one.

-readonly

Opens the specified external error data file in read-only mode. However, if you previously opened the file in read/write mode, the file remains in read/write mode.

When you enable this mode, the tool cannot modify the contents of the error data file.

The tool ignores this option when opening attached error data files and uses the open mode of the block.

The **-readwrite**, **-readonly**, and **-checkonly** options are mutually exclusive; you can specify only one.

-checkonly

Checks if the specified error data file is already open and returns a collection that contains the error data object if the file is open. If the file is not open, it returns an empty string.

The **-readwrite**, **-readonly**, and **-checkonly** options are mutually exclusive; you can specify only one.

DESCRIPTION

The **open_drc_error_data** command opens an error data file so that you can query or modify the error data. The command supports both attached error data files and external error data files.

- Attached error data files are stored in the design library.
  To open an attached error data file, specify its name or a collection that contains the error data file as a command argument.

  When you open an attached error data file, the tool always opens the file using the current block's open mode.

- External error data files are stored on disk.
  To open an external error data file, specify its file path with the **-file_name** option.

  When you open an external error data file, by default, the tool opens the file in read/write mode. To explicitly specify the open mode, use the **-readwrite** or **-readonly** option. When you use these options, the **-readwrite** option has the highest precedence. If you open the error data the **-readwrite** option and then open it again with the **-readonly** option, the open mode remains read/write.If you first open the error data with the **-readonly** option, and then open it again with the **-readwrite** option, the open mode changes to read/write.

The **open_drc_error_data** command increments the open count for the error data, except when you use the **-checkonly** option. The **open_drc_error_data** commands must be balanced by an equal number of **close_drc_error_data** commands to close the error data object.

The command returns a collection that contains the opened error data file. If no file is opened, the command returns an empty string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example opens the dppinassgn.err error data file attached to the current block and returns a collection that contains the error data file.

```
prompt> open_drc_error_data dppinassgn.err
```

```
{dppinassgn.err}
```

The following example opens the external error data file named "./my_design_dppinassgn.err" in read/write mode and returns a collection that contains the error data file.

```
prompt> open_drc_error_data -file_name ./my_design_dppinassgn.err
{my_design_dppinassgn.err}
```

SEE ALSO

close_drc_error_data(2)
collections(2)
create_drc_error_data(2)
filter_collection(2)
get_drc_error_data(2)
query_objects(2)
regexp(2)
remove_drc_error_data(2)
save_drc_error_data(2)
write_drc_error_data(2)
shell.common.collection_result_display_limit(3)

open_ems_database

Opens an existing EMS database.

SYNTAX

collection **open_ems_database**
ems_database_name

Data Types

ems_database_name string

ARGUMENTS

ems_database_name

Specifies the name of the EMS database. If the specified EMS database does not exist, the command issues an error message.

DESCRIPTION

The **open_ems_database** command opens an existing EMS database and makes it the current, active database.

EXAMPLES

The following example returns the currently open EMS database, opens the EMS database named abc.ems, then shows that abc.ems is now the current EMS database.

```
prompt> get_current_ems_database
{def.ems}
prompt> open_ems_database abc.ems
{abc.ems}
prompt> get_current_ems_database
{abc.ems}
```

SEE ALSO

close_ems_databases(2)
create_ems_database(2)
get_current_ems_database(2)
get_ems_databases(2)
save_ems_database(2)
set_current_ems_database(2)

open_lib

Open an already-existing library for edit or read access.

SYNTAX

open_return **open_lib**
[-edit | -read]
[-ref_libs_for_edit]
[-recover]
library_name

Data Types

open_return Collection containing the library or "".
library_name Library name in simple, relative, or absolute path.

ARGUMENTS

-edit

Opens the library in edit mode.

This is the default for design libraries if you do not specify any options with this command.

In an implementation tool, cell libraries can be opened only in read mode. If you try to explicitly open a cell library in edit mode, the command issues a warning message and the library is opened in read mode.

This option is mutually exclusive with the **-read** option and is not available in the library manager.

-read

Opens the library is read-only mode.

This is the default open mode for cell libraries in an implementation tool.

This option is mutually exclusive with the **-edit** option and is not available in the library manager.

-ref_libs_for_edit

Open reference module libraries in edit mode so that lower-level designs can be modified. Lib-cell libraries will still be open in read mode even if this option is used. This option can only be used when opening a design library in edit mode, ether by using the default open mode or explicitly with -edit.

To edit cell libraries, use the edit flow in the library manager (lm_shell).

This option is not available in the library manager.

-recover

Recovers the latest auto-saved library of the given library. Auto-saved library will recovered and opened as a new library and won't be merged into the current or original library.

library_name

Specifies library to open.

Because library names become directory or file names, they cannot contain the slash ("/") or colon (":") character.

RETURN VALUE

Returns the library if successful, or 0 if not. If an illegal name is given, a Tcl error is raised.

DESCRIPTION

This command opens a pre-existing library. The library can be in is distributed (file-system hierarchical) form or a single-file form. The library name can be given as a full path name, a relative path name, or a single name. In the last two cases, the search_path is used to determine the full path for the library. When the library is opened, its design catalog is read in, but no designs are read in until they are opened explicitly (through open_block) or implicitly (by being bound to from another design). As a library is being opened through this command, all of the libraries on its ref-lib list are also opened in read-only mode. These reference libraries have their design catalogs read in, but their reference libs are not read in until and unless the reference lib is also explicitly opened. The technology section in the design library, or the technology section of the dedicated technology library in the reference libraries, or the first technology section found in its reference libraries if no technology library is specified, becomes the technology data for the design library. In the library manager,if an aggregate flow is created, this command adds the cell library to an aggregate library. To edit the cell library, use the edit flow in the library manager (lm_shell).

EXAMPLES

The following command opens the design library MyDesigns.

```
prompt> open_lib MyDesigns
INFO: loading technology data from "MyLibCells"
1
prompt> get_libs
{"MyDesigns"}

prompt> get_libs -all
{"MyDesigns" "RefLib1" "RefLib2" "MyLibCells"}
```

SEE ALSO

create_lib(2)
save_lib(2)
close_lib(2)
copy_lib(2)
move_lib(2)
current_lib(2)
get_libs(2)
set_ref_libs(2)
search_path(3)

open_rail_result

Opens the rail result data that is generated by the RedHawk Analysis Fusion rail analysis run.

SYNTAX

status **open_rail_result**
[result_name]
[-block_instance *block_instance_name*]
[-top_design *top_design_name*]
[-back_annotate]

Data Types

result_name string
block_instance_name string
top_design string

ARGUMENTS

result_name

Specifies the name of the rail result to open. The default name is "REDHAWK_RESULT".

block_instance *block_instance_name*

Specifies the full path of the block instance whose rail result data is to be shown. When specified, the **open_rail_result** command loads the rail result of the top design, crops the result by the given block instance, and then opens the cropped result. Note that the opened current design is a block design, not top design. This option must be used with the -top_design option to specify the name of the top design.

top_design *top_design_name*

Specifies the name of top design and crops the top design's rail result by the -block_instance option. The option must be used with -block_instance option.

back_annotate

Forces to reconvert RAIL_DATABASE from the RedHawk/RedHawk-SC result. This attempts to recover a previous result after a design change. Instance-based results will refresh geometry. while para-based results will not refresh geometry. Newly created design elements will not have any result.

DESCRIPTION

This command opens the rail result data that is generated by the **analyze_rail** command. With the rail result data loaded, you can then check the RedHawk Analysis Fusion analysis results by displaying maps or querying attributes.

The rail results are saved in the RAIL_DATABASE directory inside the RedHawk work directory. To list the available rail results, use the **report_rail_result** command.

You can use the **set_app_options -name rail.database -value _rail_database_path_** command to switch to a different RAIL_DATABASE, but not while a rail result is already open.

If you want to debug voltage drop issues in the block design with only rail analysis result data for the top design, use the **-block_instance** and **-top_design** options to crop the top design's rail result for the specified block instance. You can then display maps or query attributes based on the cropped result for the specified block instance.

EXAMPLE

The following example opens the rail result data with default result name.

```
prompt> open_rail_result
1
```

Assume the current design name is CORE and it's instantiated in the top design by instance name u_core_1, the following example opens the top design's rail result on u_core_1 only.

```
prompt> open_rail_result -top_design top -block_instance u_core_1
1
```

The following example force reconverts RAIL_DATABASE from RedHawk/RedHawk-SC result.

```
prompt> open_rail_result ... -back_annotate
1
```

SEE ALSO

analyze_rail(2)
report_rail_result(2)
list_rail_results(2)
close_rail_result(2)
set_app_options(2)

optimize_dft

Performs timing-aware placement based or clock-aware scan reordering.

SYNTAX

status **optimize_dft**
  [-clock_aware]

ARGUMENTS

-clock_aware

Invokes clock-aware scan reordering, which aims to minimize the number of clock buffer crossings in the scan chain for better hold violation reduction along the scan path.

DESCRIPTION

The **optimize_dft** command performs scan chain reordering. By default, it performs timing friendly placement based reordering. It aims to reduce the scan chain wire length, minimize the setup timing violation along scan path, benefit congestion, and improve routability. Prior to using this command you must enable scandef into has been read into the design, and placement has been performed on the design.

When invoked with the **-clock_aware** option, the **optimize_dft** command performs clock-aware scan reordering. The reordering aims to minimize the number of clock driver crossings of the scan cells within the scan chain. Minimizing the number of clock driver crossings can reduce hold time violations in the scan chain. For best results, you should first perform timing friendly placement based scan reordering.

For best results, you should use the **place_opt** and **clock_opt** commands instead of the **optimize_dft** command. The **optimize_dft** command is provided for advanced users who want to have finer control over the placement and clock tree synthesis processes.

The standalone **optimize_dft** command does not update extraction or timing data so you must do that.

Moreover, before performing any optimization, you should run the **check_scan_chain** command because only "VALIDATED" scan chains are optimized with the **optimize_dft** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

read_def(2)
place_opt(2)
clock_opt(2)
check_scan_chain(2)
report_scan_chains(2)

optimize_rdl_routes

Optimizes the RDL routing patterns.

SYNTAX

status **optimize_rdl_routes**
[-nets *collection_of_nets* | -nets_in_file *nets_file*]
[-objects *collection_of_objects*]
-layer *tech_layer_name*
[-reserve_power_resources true | false]

Data Types

collection_of_nets collection
nets_file string
collection_of_objects collection
tech_layer_name string

ARGUMENTS

-nets ***collection_of_nets***

Specifies the nets to be optimized. By default, all RDL nets are optimized.

-nets_in_file ***nets_file***

Specifies the name of a file that contains the list of nets to be optimized.

-nets has higher priority than **-nets_in_file**.

-objects ***collection_of_objects***

Specifies the RDL nets (see **-nets**) by cells or pins. Only the sub-nets which involve the cells or pins are affected.

If the app option **flip_chip.route.routing_style** is **spanning_tree**, the whole nets of the cells or pins are regarded as involved.

-layer ***tech_layer_name***

Specifies the metal layer on which to optimize the RDL wires. You specify the metal layer by using its technology file layer name.

This option is required.

Use the **create_routing_rule** command to specify the required width and spacing of the thin nets before using this function.

-reserve_power_resources true | **false**

Specifies whether to create additional routing channels for power routing. When this option is true, the command reserves power

resources by pushing RDL routing wires to create extra routing channels between the bump cells. After running the command with this option, the amount of 'U' and 'Z' routing shapes might increase. By default, the option is false and the command does not create additional routing channels.

DESCRIPTION

This command improves and enhances the RDL routing patterns by removing and reducing the amount of 'U' and 'Z' routing shapes in the design. In order to get better optimization result, floating or dangling RDL routes will be removed to gain more space for optimizing completed routed RDL paths. To keep the floating or dangling RDL routes, please change 'shape_use' to 'user_route' before executing this command.

RDL nets are nets in flip-chip designs or 3DIC designs which are routed on the redistribution layer. In a flip-chip design, RDL nets connect a bump cell to driver I/O cells. In a 3DIC or interposer design, RDL nets can connect the following elements:

- A micro-bump cell and a micro-bump cell
- A micro-bump cell and a TSV (through silicon via) cell
- A micro-bump cell and driver I/O cell

Micro-bump cells are also called flip-chip pads.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example optimizes the RDL routing patterns on the METAL9 layer.

```
prompt> optimize_rdl_routes -layer METAL9
```

The following example optimizes the RDL routing patterns in NET1 and NET2 nets on the METAL7 layer.

```
prompt> optimize_rdl_routes -layer METAL7 -nets {NET1 NET2}
```

SEE ALSO

push_rdl_routes(2)
route_rdl_flip_chip(2)
create_routing_rule(2)

optimize_routability

Optimizes the routability of a design.

SYNTAX

status **optimize_routability**
[-drc_rules *drc_rules_list*]
[-layer_rules *layer_list*]
[-keepout_width *width*]
[-flip]
[-route]
[-remove_keepouts]
[-check_drc_rules]

Data Types

drc_rules_list list
layer_list list

ARGUMENTS

-drc_rules *drc_rules_list*

Specifies the drc rules to work on. This should be a list consisting of one or more lists of the name of the drc rule and its corresponding layer. See an example of usage below. If this option or *-layer_rules* option is not specified, then all drc rules are considered.

-layer_rules *layer_list*

Specifies the layer names to work on. If this option or *-drc_rules* is not specified, then all drc rules are considered.

-keepout_width *width*

Specifies the keepout width to use, in microns. The default value is the site width.

-flip

Specifies that orientation changes are performed instead of spacing cells apart.

-route

Performs **route_eco** command and **check_routes** command. The route option used is **modified_nets_first_then_others.**

-remove_keepouts

Remove all optimize_routability inserted keepouts. This can be performed at the end of optimize_routability.

-check_drc_rules

Print information about DRC rules. If this option is specified, other options will be ignored. The design remains unchanged.

DESCRIPTION

The **optimize_routability** command helps to improve routability of a design by spacing cells further apart with the use of keepout_margins or by flipping cells. User should run **check_routes** command before using this command to ensure that the DRC error information is up-to-date.

EXAMPLES

The following example runs the **optimize_routability** command using specified DRC rules.

```
prompt> optimize_routability -drc_rules {{ "Local double pattern cycle" "M2" }
{ "Diff net spacing" "{M2  M2}" } { "Diff net via-cut spacing" "{VIA1  VIA2}" }
```

SEE ALSO

get_drc_error_data(2)
get_drc_errors(2)

optimize_routes

Optimizes the routing to improve the routing patterns and increase the yield during manufacturing.

SYNTAX

status **optimize_routes**
[-max_detail_route_iterations *num*]
[-nets *list* [-reroute_all_shapes_in_nets true | false]]

Data Types

num integer
list list

ARGUMENTS

-max_detail_route_iterations *num*

Specifies the number of search and repair loops to run after optimization.

Router runs the specified number of detail route iterations if using **set_app_options -list {route.detail.force_max_number_iterations true}** command.

The default value is 40. The range is 0 to 1000.

-nets *list*

Specifies the nets to optimize.

If you do not specify this option, router selects a group of nets to optimize based on their routing patterns and the value from the **route.detail.optimize_wire_via_effort_level** application option.

-reroute_all_shapes_in_nets true | false

Controls whether all shapes on the specified nets are rerouted by the command, regardless of the existing routing patterns.

The option has an effect only when used with the **-nets** option; it is ignored if you do not use the **-nets** option.

The default value is false.

DESCRIPTION

This command performs detail route optimization. It detects nonoptimal routing patterns on signal nets and clock nets and tries to

reroute them to improve the yield and minimize the wire length and via count. Router picks the nets to optimize based on their routing patterns and the optimize wire and via effort level, as specified by the **route.detail.optimize_wire_via_effort_level** application option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example performs three search and repair loops after optimization.

```
prompt> optimize_routes -max_detail_route_iterations 3
1
```

The following example optimize nets N1, N2, and N3 by rerouting all of their shapes.

```
prompt> optimize_routes -nets {N1 N2 N3} \
     -reroute_all_shapes_in_nets true
```

SEE ALSO

route_detail(2)
route.detail.optimize_wire_via_effort_level(3)

optimize_topology_plans

Optimizes topology plans in the current block by creating routing topologies, pins, and feedthroughs based on command options.

SYNTAX

collection **optimize_topology_plans**
[-mode optimize_only | derive_topology_only | optimize_and_derive_topology | repeaters_only]
[-routing_style topology_plan_router | no_route | topology_router]
[*topology_plan_list*]

Data Types

topology_plan_list collection

ARGUMENTS

topology_plan_list

Specifies a list of topology plans to optimize or derive. The list can contain topology plan names, patterns, or collections. A collection can be specified by using the **get_topology_plans** command. When no topology plan is provided, the command will collect all the topology plans in the current design to optimize.

-mode optimize_only | derive_topology_only | optimize_and_derive_topology | repeaters_only

Direct the command to perform optimization only or derive topology as follows:

- optimize_only: Create feedthroughs, place pins.
- derive_topology_only: Derive the new topology plan based on global routing results. New topology nodes will be created based on the global routing location at the boundary of physical blocks. New topology edges will created to connect those new nodes based on global routing results. In addition, insert virtual register/buffer locations per net estimation rule. The command doesn't modify netlist in the design, i.e., no feedthroughs or pins will be created.
- optimize_and_derive_topology: Create feedthroughs and pins based on global routing results. In addition, topology nodes/edges will be created based on the new created feedthroughs/pins and virtual registers/buffers will be inserted based on the net estimation rule.
- repeaters_only: Insert virtual register/buffer locations per net estimation rule. The command doesn't add new topology nodes or edges.

-routing_style topology_plan_router | topology_router | no_route

Direct the command to perform optimization with specified routing mode as follows. By default, the topology_plan_router will be used.

- topology_plan_router: Use the global router to route the topology plan and generate one set of shape layer list to represent

whole routing together. The global router will consider the demand for all the associated nets when generating the topologies for the plan.

- topology_router: Route the nets separately that are associated with plan. It may result in increased runtime due to more net routings.
- no_route: Create feedthroughs and pins based on the plan definition. In order to create feedthroughs in no_route mode, users need to define the topology plan with complete routing topology. For instance, users will need to specify the blocks that the topologies going through. If the plan definition missed feedthrough blocks, single pins maybe created in fully abutted designs.

DESCRIPTION

The **optimize_topology_plans** command derives routing topologies based on user specified topology plans using global routing. The new block pins or feedthroughs might be created using this command. The command will create routing shape layer list for each topology edge and insert virtual registers and buffers based on net estimation rule defined in topology plan. The new created shape layer list can be used to guide detail route in later planning stage. The virtual register/buffer can be used to guide later real register/buffer insertion.

EXAMPLES

The following example optimize all the topology plans using topology plan router in the current block:

```
prompt> optimize_topology_plans
```

The following example optimize topology plan with name "myplan" with no_route style to create feedthroughs/pins without global routing:

```
prompt> optimize_topology_plans [get_topology_plans myplan] -routing_style no_route
```

SEE ALSO

check_topology_plans(2)
create_topology_plan(2)
current_topology_plan(2)
report_topology_plans(2)
write_topology_plans(2)

parallel_execute

Executes a list of read-only commands in parallel.

SYNTAX

status **parallel_execute**
[-commands_only]
[-list_allowed_commands]
[-max_cores *count*]
[*command_list*]

Data Types

count integer
command_list string

ARGUMENTS

-commands_only

Specifies that all commands listed in the *command_list* argument are specified without a log file. The output of every command is written out after all commands have completed.

-list_allowed_commands

Lists all the allowed read-only commands supported by the **parallel_execute** command.

This option cannot be used with any other options or arguments.

-max_cores *count*

Specifies the maximum number of commands that can be executed concurrently at any given time. You can specify any positive integer greater than or equal to 1.

The default for this option is as specified by the **set_host_options -max_cores** command. If the **set_host_options -max_cores** command was not run, the default is 1.

When the maximum number of cores is 1, the command runs the specified commands sequentially.

If you use the **parallel_execute** command within the **redirect -bg** command, the specified value must be less than the value specified with the **-max_cores** option of the **redirect -bg** command. Otherwise, the maximum number of cores for the **parallel_execute** command is reduced to the maximum number of cores specified for the **redirect -bg** command.

command_list

Specifies the list of commands to execute in parallel. Use the following syntax to specify each command to run and the log file for

its output:

{*command*} *log_file*

Each command must be specified on its own line. If the command has no options, the braces {} are optional.

Instead of creating a log file for each command, you can output the results for all of the commands to the console after all the commands complete by using the **-commands_only** option. In this case, do not specify a log file for each command.

DESCRIPTION

The **parallel_execute** command provides a mechanism to easily execute a set of read-only commands in parallel on the same machine where the main process is running. The spawned commands access the same in-memory data for the same design. The parent process is blocked until the longest running command completes. To run the commands in parallel in the background, run the **parallel_execute** command with the **redirect -bg** command.

The commands must be independent of each other, must not change the design database, and are not inherently multithreaded, thereby saving time on their sequential execution. User-defined procedures can be executed; however, they must use only supported commands. The commands supported by the **parallel_execute** command are primarily reporting and querying commands. To list the supported commands, use the **-list_allowed_commands** option. Nested **parallel_execute** commands are not allowed.

If you specify unsupported commands, the **parallel_execute** command issues an error message and continues executing the remaining commands.

When you use this command consider the following guidelines:

- Some reporting commands perform a timing update internally if the design timing is not up-to-date. If these commands are executed in parallel, each command runs the timing update independently.

  To reduce runtime and memory usage, run the **update_timing** command before running the **parallel_execute** command.

- The spawned commands cannot pass variables back to the parent process.

  To pass variables from a child process to the parent process, you must write the contents of the variables to a file during the child process, and then read that file in the parent process.

- The **parallel_execute** command implicitly links the design if needed.

The **parallel_execute** command itself does not check out any license keys. The commands specified within **parallel_execute** check out licenses as required. A license key checked out in this way can be reused by all the other commands inside the **parallel_execute**. If the required keys are already checked out when you run the **parallel_execute** command, those keys are reused by the child commands inside **parallel_execute**, and no extra keys are checked out. If a command within **parallel_execute** checks out a license on its own, it checks it back in at the end of its invocation. Thus the status of licenses checked out before and after **parallel_execute** are the same.

Each spawned command is run in a single core. If the **parallel_execute** command fails to spawn a command, it executes the command sequentially after all other spawned commands have completed.

In the following cases, the **parallel_execute** command runs the commands sequentially:

- There is only one command inside **parallel_execute**.
- The number of commands to run in parallel is set to 1 with the **-max_cores** option.

In these cases, there is not any parallelization. Single threading is not enforced and the number of licenses checked out is proportional to the **set_host_options -max_cores** value. If a command within **parallel_execute** checks out a license on its own, the

license is not checked back in for these special cases.

EXAMPLES

The following example lists all allowed read-only commands.

```
prompt> parallel_execute -list_allowed_commands

Allowed read-only commands are:
- cmd1
- cmd2
 ...
```

The following example executes a single command and writes the output to the log file named log1. The command is directly executed, not spawned.

```
prompt> parallel_execute {
? {report_clock_qor -clocks [get_clocks test*]} log1
? }
? ? Only one command specified, running it in serial manner.
Running command : 'report_clock_qor -clocks [get_clocks test*]' > log1

All command runs are done. Below is the status.

Completed 'report_clock_qor -clocks [get_clocks test*]' successfully.
Output is in log 'log1'
```

The following example executes three commands by spawning them and writing out their output. The maximum number of concurrent commands is 5 as set by the **set_host_options** command.

```
prompt> set_host_options -max_cores 5
prompt> parallel_execute -commands_only {
? report_cells
? {report_site_defs -nosplit}
? {report_delay_calculation -from pin1 -to pin2}
? }
Dispatching command : 'report_cells'
Dispatching command : 'report_site_defs -nosplit'
Dispatching command : 'report_delay_calculation -from pin1 -to pin2'
Waiting for all dispatched commands to complete ...
Done.

All command runs are done. Below is the status.

Completed 'get_cells' successfully.
Output given below:

****************************************
Report : cell
 ...

Completed 'report_site_defs -nosplit' successfully.
Output given below:
```

```
****************************************
Report : report_site_defs
 ...

Completed 'report_delay_calculation -from pin1 -to pin2' successfully.
Output given below:

****************************************
Report : delay_calculation
 ...
```

The following example is the same as the previous example except that only two commands can run in parallel.

```
prompt> parallel_execute -commands_only -max_cores 2 {
? report_cells
? {report_site_defs -nosplit}
? {report_delay_calculation -from pin1 -to pin2}
? }
```

SEE ALSO

redirect(2)
set_host_options(2)

parse_proc_arguments

Parses the arguments passed into a Tcl procedure.

SYNTAX

string **parse_proc_arguments** -args *arg_list*
result_array

list *arg_list*
string *result_array*

ARGUMENTS

-args *arg_list*

Specified the list of arguments passed in to the Tcl procedure.

result_array

Specifies the name of the array into which the parsed arguments should be stored.

DESCRIPTION

The **parse_proc_arguments** command is used within a Tcl procedure to enable use of the -help option, and to support argument validation. It should typically be the first command called within a procedure. Procedures that use **parse_proc_arguments** will validate the semantics of the procedure arguments and generate the same syntax and semantic error messages as any application command (see the examples that follow).

When a procedure that uses **parse_proc_arguments** is invoked with the -help option, **parse_proc_arguments** will print help information (in the same style as using **help -verbose**) and will then cause the calling procedure to return. Similarly, if there was any type of error with the arguments (missing required arguments, invalid value, and so on), **parse_proc_arguments** will return a Tcl error and the calling procedure will terminate and return.

If you didn't specify -help, and the specified arguments were valid, the array variable *result_array* will contain each of the argument values, subscripted with the argument name. Note that the argument name here is NOT the names of the arguments in the procedure definition, but rather the names of the arguments as defined using the **define_proc_attributes** command.

The **parse_proc_arguments** command cannot be used outside of a procedure.

EXAMPLES

The following procedure shows how **parse_proc_arguments** is typically used. The argHandler procedure parses the arguments it receives. If the parse is successful, argHandler prints the options or values actually received.

```
proc argHandler args {
  parse_proc_arguments -args $args results
  foreach argname [array names results] {
    echo "  $argname = $results($argname)"
  }
}
define_proc_attributes argHandler -info "argument processor" \
  -define_args \
  {{-Oos "oos help" AnOos one_of_string {required value_help {values {a b}}}}
   {-Int "int help" AnInt int optional}
   {-Float "float help" AFloat float optional}
   {-Bool "bool help" "" boolean optional}
   {-String "string help" AString string optional}
   {-List "list help" AList list optional}}
  {-IDup int dup AIDup int {optional merge_duplicates}}}
```

Invoking argHandler with the -help option generates the following:

```
prompt> argHandler -help
Usage: argHandler    # argument processor
       -Oos AnOos            (oos help:
                             Values: a, b)
       [-Int AnInt]          (int help)
       [-Float AFloat]       (float help)
       [-Bool]               (bool help)
       [-String AString]     (string help)
       [-List AList]         (list help)
```

Invoking argHandler with an invalid option causes the following output (and a Tcl error):

```
prompt> argHandler -Int z
Error: value 'z' for option '-Int' not of type 'integer' (CMD-009)
Error: Required argument '-Oos' was not found (CMD-007)
```

Invoking argHandler with valid arguments generates the following:

```
prompt> argHandler -Int 6 -Oos a
  -Oos = a
  -Int = 6
```

SEE ALSO

define_proc_attributes(2)
help(2)
proc(2)

partition_block

Create a partitioned version of the current block based on a physical region.

SYNTAX

int **partition_block**
[-region *region*]
[-halo \fhalo]
object

Data Types

region polyrect
halo distance

ARGUMENTS

-region *region*

A required argument that specifies a rectilinear boundary from which to create a single partition block. Accepts both bbox and bounary coordinate list specifiers.

-halo *halo*

An optional argument that specifies a halo or context buffer around each partition region. When given, each partition region extends by the halo distance on each side.

DESCRIPTION

The **partition_block** command creates a single clipped or partitioned block that captures a given region of the current block. The command saves the resulting partition block into a new library. The library has the name <current_lib>_partition_block, and if a library exists with the same name (from previous calls to **partition_block**) then a numerical suffix is appened to create a unique name. The partitioned block keeps the name of its source block.

The command accepts a rectilinear region, and an optional halo region. The effective partition area is the given region, exteneded on all sides by the halo value if specified. The resulting partition block contains the physical objects that intersect the partition area, along with the corresponding pieces of the netlist. A partition block is always a valid design, though not always a complete sub-circuit. The partition operation makes no additional edits to clean up incomplete states such as dangling nets, missing drivers, etc. Rather the partition block aims to be a most accurate copy of only the data present in the given partition region.

EXAMPLES

The following command creates a partition block using a bbox region

prompt> **partition_block -region {{0 0} {10 10}}**

The following example creates a partition block using a rectilinear region

prompt> **partition_block -region {{5 5} {5 100} {100 100} {100 95} {10 95} {10 5}}**

The following example creates a partition block with a halo

prompt> **partition_block -region {{0 0} {20 20}} -halo 2**

SEE ALSO

save_block(2)
copy_block(2)

place_eco_cells

Performs coarse placement and legalization on ECO cells in the design.

SYNTAX

status **place_eco_cells**
-cells *cell_objects* | -unplaced_cells | -eco_changed_cells
[-no_legalize]
[-legalize_only]
[-channel_aware]
[-legalize_mode free_site_only | allow_move_other_cells | minimum_physical_impact]
[-displacement_threshold *threshold*]
[-max_displacement_threshold *max_distance*]
[-fixed_connection_net_weight *weight*]
[-honor_user_net_weight]
[-use_virtual_connection]
[-max_fanout *max_fanout*]
[-ignore_pin_connection *pin_name_list*]
[-remove_filler_references *references*]

Data Types

cell_objects collection
threshold float
max_distance float
weight integer
max_fanout integer
pin_name_list list

ARGUMENTS

-cells *cell_objects*

Places the group of specified cells. You must specify one of the *-cells*, *-unplaced_cells* and *-eco_changed_cells* options. These options are mutually exclusive with each other.

-unplaced_cells

Places all unplaced cells. You must specify one of the *-cells*, *-unplaced_cells* and *-eco_changed_cells* options. These options are mutually exclusive with each other.

-eco_changed_cells

Places all the cells with the **eco_change_status** attribute with values **create_cell**, **add_buffer**, **change_link**, **size_cell**, or

add_buffer_on_route. You must specify one of the *-cells*, *-unplaced_cells* and *-eco_changed_cells* options. These options are mutually exclusive with each other.

-no_legalize

Runs placement without legalizing the cells. Use this option when you must run **place_eco_cells** multiple times for different purposes, but legalization needs to be done only one time. By default, this option is off.

-legalize_only

Performs only legalization without placement on the user-provided cells, if you have decided the cell locations. This option is used only with the *-cells* or *-eco_changed_cells* option. The *-legalize_only* and *-no_legalize* options are mutually exclusive.

- If you specify the *-legalize_only -cells* option, the command applies to cells that already have locations, that is, the **is_placed** attribute is TRUE. When the option is specified, you need to supply locations for all input cells. If there is an unplaced cell, the command fails with a EPL-020 error.
- If you specify the *-legalize_only -eco_changed_cells* option, the command applies to cells with the **eco_change_status** attribute that has values **create_cell**, **add_buffer**, **change_link**, **size_cell**, or **add_buffer_on_route**, and the **is_placed** attribute as TRUE. The command skips the cells that are not placed with the warning message EPL-021.

  To perform only legalization without placement for the cells which have **is_placed** attribute set to false, you can change the cell status with the **set_attribute $cells physical_status placed** command before running **place_eco_cells**. Note that you need to make sure to supply locations for these cells, otherwise, you might get unexpected results.

-channel_aware

Uses the channel area between nonabutted macro for ECO placement. For example, buffer chain or pipeline registers can be placed by using those channel areas to produce a better placement result. By default, this option is off.

-legalize_mode free_site_only | allow_move_other_cells | minimum_physical_impact

Specifies the legalization mode. The default is **free_site_only**. This option cannot be used with *-no_legalize*. *-legalize_mode minimum_physical_impact* can only be used with *-legalize_only*.

- **free_site_only**: The ECO cells are legalized on free sites without moving other existing cells.
- **allow_move_other_cells**: The ECO cells are legalized on the nearest legal locations with moving other existing cells.
- **minimum_physical_impact**: Combine ECO cell legalization (free sites only and push non-ECO cell away) into a single, minimum physical impact legalization. Legalize ECO cells on free sites without moving other existing cells, and then do legalization with moving other existing cells for cells with large displacement. This allows non-ECO cells to be pushed away to make space for ECO cells.

-displacement_threshold *threshold*

Specify the displacement threshold value. The units are microns and the value cannot be negative. This option cannot be used together with the *-no_legalize* option. This option cannot be used together with *-legalize_mode allow_move_other_cells*. When this option is used in different legalize mode, the behavior is different. The command will not legalize the cells whose legalization displacement will exceed the specified threshold. That is, the tool keeps the locations of those cells before legalization. And the tool reports the total number of the rejected cells due to the specified threshold and creates a collection *epl_legalizer_rejected_cells* to track these rejected cells. If the collection is already there before **place_eco_cells**, it will be reset. For the cells in the epl_legalizer_rejected_cells collection, you can use *legalize_placement* to push the non-ECO cells to make space for the ECO cells.

- **free_site_only**: The command will not legalize the cells whose legalization displacement will exceed the specified threshold. That is, the tool keeps the locations of those cells before legalization. And the tool reports the total number of the rejected cells due to the specified threshold and creates a collection called *epl_legalizer_rejected_cells* to track these rejected cells. If the collection is already there before running **place_eco_cells**, it will be reset. For the cells in the epl_legalizer_rejected_cells collection, you can use the *place_eco_cells -legalize_mode allow_move_other_cells -*

legalize_only command to push the non-ECO cells to make space for the ECO cells.

- **minimum_physical_impact**: The command first legalizes cells on free sites, but does not legalize the cells whose legalization displacement will exceed the specified threshold. And then collect these cells to do legalization with moving other existing cells again.

-max_displacement_threshold *max_distance*

Specify the maximum displacement threshold value. The units are microns and the value cannot be negative. This option must be used with the *-displacement_threshold* option. The threshold specified by the *-max_displacement_threshold* option should be greater than or equal to the threshold specified by the *-displacement_threshold* option. When this option is used in different legalize mode, the behavior is different. The command creates a collection called *epl_max_displacement_cells* to track the ECO cells which have a legalization displacement that exceeds the threshold specified by the *-max_displacement_threshold* option. If the collection already exists before this command run, it will be reset. The command also does not legalize the cells since the displacement of these cells also exceeds the threshold specified by the *-displacement_threshold* option. The collection *epl_max_displacement_cells* is a subset of the collection *epl_legalizer_rejected_cells*. For the cells in the collection *epl_max_displacement_cells*, you can decide to reject the ECO changes for these larger displacement ECO cells.

- **free_site_only**: The command creates a collection called *epl_max_displacement_cells* to track the ECO cells which have a legalization displacement that exceeds the threshold specified by the *-max_displacement_threshold* option. If the collection already exists before this command run, it will be reset. The command also does not legalize the cells since the displacement of these cells also exceeds the threshold specified by the *-displacement_threshold* option. The collection *epl_max_displacement_cells* is a subset of the collection *epl_legalizer_rejected_cells*. For the cells in the collection *epl_max_displacement_cells*, you can decide to reject the ECO changes for these larger displacement ECO cells.
- **minimum_physical_impact**: The command first legalizes cells on free sites. If cells exist whose legalization displacement will exceed the specified max threshold, the legalization will failed. And the *epl_legalizer_rejected_cells* and *epl_max_displacement_cells* collections will be created to record cells according to their displacement as if the "on free site" legalization happened.

-fixed_connection_net_weight *weight*

Specify the net weight for net connected to fixed point, such as the I/O pad/macro. The value is integer type and should be great than or equal to 2. The option cannot be used with *-legalize_only* and *-use_virtual_connection*.

The weight of net connected to the fixed point is set to the given value, while the weight of other nets is set to default value 1. **place_eco_cells** does coarse placement according to user-specified fixed connection net weight, and places the ECO cell close to the fixed point to which it connects directly. The net weight determines how close the ECO cell is to the fixed point.

The option can support virtual connection-based channel-aware coarse placement when you also specify *-channel_aware*.

-honor_user_net_weight

Performs placement based on user specified net weight setting. The option works together with two commands to create and get the user specified net weight: **set_eco_placement_net_weight** and **report_eco_placement_net_weight**.

The option cannot be used with *-legalize_only* and *-fixed_connection_net_weight*. The option can support channel-aware coarse placement when it used with the *-channel_aware* option.

User needs to set net weight by set_eco_placement_net_weight before place_eco_cells -honor_user_net_weight, otherwise, the command will error out.

-use_virtual_connection

Performs placement based on user-defined virtual connection. The option works together with a set of commands to create, modify and get the virtual connection: **create_virtual_connection**, **add_pins_to_virtual_connection**, **remove_pins_from_virtual_connection**, **remove_virtual_connections**, **"set_attribute -objects virt_conn_list -name name|weight|pins -value $value"**, **"get_attribute -objects virt_conn_list -name name|weight|pins"** and **get_virtual_connections**.

The option cannot be used with *-legalize_only* and *-fixed_connection_net_weight*. The option can support virtual connection-based channel-aware coarse placement when it used with the *-channel_aware* option.

This option must be used with the *-cells* option. The given cell must have at least one virtual driver and one virtual load. Such input cells are considered to be movable cells and placed by the **place_eco_cells** command. The given cell with only input or output virtual connection or without virtual connection will be considered as fixed cells and skipped to do **place_eco_cells** even if the cell exists in cell list of *-cells* option. The command issues a warning for such cells. If all given cells are skipped, the command errors out with no valid cell to be placed.

-max_fanout *max_fanout*

Specify the maximum fanout value to use when ignoring high-fanout net connections of the ECO cells being placed. The command ignores any net connected to the ECO cell if its fanout exceeds the maximum fanout specified. Since the placement of an ECO cell depends on its connectivity, the coarse placement result is affected by this option. However, there is no impact on legalization. If this option is used with the *-legalize_only* option, the command fails with a CMD-001 error message. If you do not specify this option, the tool uses a default of 100 for the maximum fanout.

-ignore_pin_connection *pin_name_list*

Specify the list of pin names, the connections to which should be ignored when placing ECO cells. The pin name must be the reference cell pin name. The pin can be on an ECO cell or on a fixed cell. The command ignores the connection of the specified pins for both eco cells and fixed cells. Since the placement of an ECO cell depends on its connectivity, the coarse placement result is affected by this option. However, there is no impact on legalization. If this option is used with the *-legalize_only* option, the command fails with a CMD-001 error message.

-remove_filler_references *references*

Specify the references of filler cells which can be ignored during Eco legalization. Unfixed filler cells of the specified references will be ignored and Eco cells can be placed in locations that overlap with these filler cells. And at the end of legalization, those filler cells overlapped with other cells will be removed. Other filler cells that not of the specified references are still treated as normal cells and cannot not be ignored during Eco legalization.

Fixed filler cells are always treated as normal cells and cannot be ignored even if their references are specified.

Since some filler cells may be removed, you may need to reinsert smaller filler cells to fill the space.

This option supports all three legalization modes. This option are mutually exclusive with -no_legalize.

DESCRIPTION

This command performs coarse placement and legalization on ECO cells in the design. ECO cells are a small number of cells that are passed into the command as a collection, or a small number of unplaced cells as a result of the **eco_netlist** or Tcl editing command runs. They can also be a small number of cells with the **eco_change_status** attribute that has the values **create_cell**, **add_buffer**, **change_link**, **size_cell**, or **add_buffer_on_route**. The attribute is set by the user or as a result of the **eco_netlist** or Tcl editing command runs.

This command legalizes the cells only on unoccupied sites in the neighborhood of the ECO cell location. The output of this command is a legally placed netlist. This command is designed for connectivity-based incremental ECO placement and provides the following features:

Ease of Use

This command places only the ECO cells and leaves other existing cells untouched to minimize the impact to the existing cell placements. It uses only free sites and does not push other cells away. You do not need to set other cells fixed.

The placement and legalization results from this command are user-controllable. You can select which cells for placement with

the *-cells* option and legalize the cells with user-specified locations using the *-legalize_only* option.

Runtime

The runtime is far less than for full-chip placement.

QoR

The placement result is connectivity-aware, blockage-aware and macro- aware. After obtaining the initial location from the connectivity, the command adjusts the location to avoid the blockage or macro if needed.

In general, the following cells are not considered as ECO cells. If they are specified, the command ignores them and issues an EPL-003 warning message:

- Hierarchical cell
- Fixed cell
- Macro cell
- Filler cell
- I/O pad cell
- Spare cell
- Physical only cell
- Cell without a signal net connection. Cells that connect only to tie nets are more like spare cells.

Using with the eco_change_status Attribute

By default, running the **place_eco_cells** command with or without the *-legalize_only* option resets the **eco_change_status** attribute to **eco_legalized**. However, when you run the command with the *-no_legalize* option, the **eco_change_status** attribute is reset to **eco_placed** for the eco_changed cells. For non eco_changed cells, the **eco_change_status** attribute remains the same. This reset operation is only needed when you legalize the ECO changed cells.

Legalization displacement report

During legalization, the tool reports legalization displacement summary as below.

```
****************************************
  Report : ECO Legalize Displacement
  Design : CORE
****************************************
avg cell displacement:    2.590 um ( 0.60 row height)
max cell displacement:    3.500 um ( 0.81 row height)
std deviation:          0.997 um ( 0.23 row height)
number of cell moved:       2 cells (out of 3 cells)
```

Flow usage about user specified net weight based coarse placement

```
open_block CORE
set_eco_placement_net_weight -nets $nets1 weight 2
set_eco_placement_net_weight -nets $nets2 weight 3

/* $net1 with weight 2, $net2 with weight 3, the others with default weight 1 */
place_eco_cells -cells $cellList -honor_user_net_weight
```

```
report_eco_placement_net_weight -output $file_name

close_block
open_block CORE

/* load net weight setting file to memory */
source $file_name

/* change weight for $nets1 */
set_eco_placement_net_weight -nets $nets1 -weight 3

/* $net1 with weight 3, $net2 with weight 3, the others with default weight 1 */
place_eco_cells -honor_user_net_weight

/* reset all net weight */
set_eco_placement_net_weight -reset
```

Flow usage about virtual connection based coarse placement

- Step 1: Prepare the script to define virtual connection / netlist for target ECO cells. See the man page for **create_virtual_connection**. The target cell must have at least one virtual connection on both input and output pins.
- Step 2: Source the predefined script. The tool loads into memory the virtual connection based on the script.

  (for example, **source my_virtual_connection.tcl**)
- Step 3: Choose the target ECO cells on which to do coarse placement based on the virtual connection. The *num_of_virtual_connection* attribute records number of virtual connections to the ECO cell. The **create_virtual_connection**, **add_pins_to_virtual_connection**, **remove_pins_from_virtual_connection**, **remove_virtual_connections** and **"set_attribute -objects virt_conn_list -name name|weight|pins -value $value"** commands update the attribute value.

  (for example, **set theCells [get_cells -filter num_of_virtual_connection != 0 -hierarchical]**)
- Step 4: The **place_eco_cells** command builds the connection network and does coarse placement based on pure virtual connection. The command also does legalization. The command will not mix the virtual connection with real connection when building connectivity.

  (for example, **place_eco_cells -cells $cells -use_virtual_connection place_eco_cells -cells $cells -use_virtual_connection -no_legalize**)
- Step 5: Manually clean up the in-memory virtual connection data after cells are placed. The tool will clean up the in-memory virtual connection data automatically when the tool quits. When the tool is restarted, you must source the predefined script again to load the virtual connection data before running the **place_eco_cells** command.

  (for example, **remove_virtual_connections -all**)

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples show how to run the **place_eco_cells** command with different cell specifications.

```
prompt> place_eco_cells -cells {STACK_BLK/NEW1 STACK_BLK/NEW2}
```

prompt> **place_eco_cells -unplaced_cells**

prompt> **place_eco_cells -eco_changed_cells**

The following examples show how to use the **-no_legalize** option.

prompt> **place_eco_cells -unplaced_cells -no_legalize**

prompt> **place_eco_cells -cells {STACK_BLK/NEW1 STACK_BLK/NEW2} -no_legalize**

prompt> **place_eco_cells -eco_changed_cells -no_legalize**

The following examples show how to use the **-legalize_only** option.

prompt> **place_eco_cells -cells {STACK_BLK/NEW1 STACK_BLK/NEW2} -legalize_only**

prompt> **place_eco_cells -eco_changed_cells -legalize_only**

The following examples show how to use the **-channel_aware** option.

prompt> **place_eco_cells -channel_aware -cells {STACK_BLK/NEW1 STACK_BLK/NEW2}**

The following examples show how to use the **-legalize_mode** option.

prompt> **place_eco_cells -cells {STACK_BLK/NEW1 STACK_BLK/NEW2} -legalize_mode free_site_only**

prompt> **place_eco_cells -cells {STACK_BLK/NEW1 STACK_BLK/NEW2}** \
-legalize_mode allow_move_other_cells

prompt> **place_eco_cells -cells {STACK_BLK/NEW1 STACK_BLK/NEW2}** \
-legalize_mode allow_move_other_cells -legalize_only

prompt> **place_eco_cells -cells {STACK_BLK/NEW1 STACK_BLK/NEW2}** \
-legalize_mode minimum_physical_impact -legalize_only

The following examples show how to use the **-displacement_threshold** option.

The cells for which the legalization displacement exceeds the specified threshold 10 micron are not legalized. The tool keeps the cell location before legalization and the cells are not moved.

prompt> **place_eco_cells -cells $eco_cells -legalize_only -displacement_threshold 10**

Example output:

```
Total 5 cells are not legalized since the legalization displacement exceeds
the specified threshold (e.g. > 10.000 um or 2 row height)
```

The cells for which the legalization displacement exceeds the specified threshold of 8.5 microns are not legalized. The tool keeps the cell location before legalization and the cells are moved to the location after coarse placement by **place_eco_cells**.

prompt> **place_eco_cells -cells $eco_cells -displacement_threshold 8.5**

The following examples show the large displacement report.

prompt> **set legalize_displace_print_count_limit 20**
prompt> **set legalize_print_displacement_threshold 3**
prompt> **place_eco_cells -cells $eco_cells**

Example output:

```
 Large displacement cells:
  Cell: CNTL_BLK/NEW2 (NAND2X1.FRAM)
```

```
    Input location: (0.000 0.000)
    Legal location: (1.680 12.960)
    Displacement: 13.068 um, e.g. 3.03 row height.
  Cell: CNTL_BLK/NEW3 (NAND2X1.FRAM)
    Input location: (0.000 0.000)
    Legal location: (3.360 12.960)
    Displacement: 13.388 um, e.g. 3.10 row height.

  Total 2 cells has large displacement (e.g. > 12.960 um or 3 row height)
```

The following examples uses the **-max_displacement_threshold** option. The cells which legalization displacement exceeds the specified threshold 10 micron are not legalized. The tool creates two collections: epl_legalizer_rejected_cells and epl_max_displacement_cells.

```
prompt> place_eco_cells -cells $eco_cells -legalize_only \
  -displacement_threshold 10 -max_displacement_threshold 20
```

Example output:

```
Total 5 cells are not legalized since the legalization displacement exceeds
the specified threshold (e.g. > 10.000 um or 2 row height)

Total 3 cells are physical-unfriendly ECO cells which displacement exceeds
the user specified threshold distance (e.g. > 20.000 um or 4 row height)
```

The following examples use the **-displacement_threshold** option with **-legalize_mode minimum_physical_impact**.

The cells for which legalization displacement exceeds the specified threshold 10 micron in "on free site" legalization will be legalized again with pushing non-ECO cells.

```
prompt> place_eco_cells -cells $eco_cells -legalize_only \
  -legalize_mode minimum_physical_impact -displacement_threshold 10
```

Example output:

```
****************************************
  Report : ECO Legalize Displacement
  Design : CORE
****************************************
avg cell displacement:    2.590 um ( 0.60 row height)
max cell displacement:    3.500 um ( 0.81 row height)
std deviation:            0.997 um ( 0.23 row height)
number of cell moved:         2 cells (out of 3 cells)
```

The following examples use the **-max_displacement_threshold** option with **-legalize_mode minimum_physical_impact**.

The legalization fails if cells whose legalization displacement exceeds the specified max threshold 10 micron in "on free site" legalization are present.

```
prompt> place_eco_cells -cells $eco_cells -legalize_only \
  -legalize_mode minimum_physical_impact \
  -displacement_threshold 5 -max_displacement_threshold 10
```

Example output:

```
Error: minimum physical impact legalization failed due to physical unfriendly cells
```

The following examples use the **-fixed_connection_net_weight** option.

```
prompt> place_eco_cells -cells $cells -fixed_connection_net_weight 10
prompt> place_eco_cells -cells $cells -fixed_connection_net_weight 10 -no_legalize
```

The following examples use the **-fixed_connection_net_weight** option with **-channel_aware**.

```
prompt> place_eco_cells -cells $cells -fixed_connection_net_weight 5 \
  -channel_aware
prompt> place_eco_cells -cells $cells -fixed_connection_net_weight 5 \
  -no_legalize -channel_aware
```

The following examples show how to use the **-honor_user_net_weight** option.

```
prompt> place_eco_cells -cells $cells -honor_user_net_weight
prompt> place_eco_cells -cells $cells -honor_user_net_weight -no_legalize
```

The following examples use the **-honor_user_net_weight** option with **-channel_aware**.

```
prompt> place_eco_cells -cells $cells -honor_user_net_weight -channel_aware
prompt> place_eco_cells -cells $cells -honor_user_net_weight -no_legalize -channel_aware
```

The following examples show how to use the **-use_virtual_connection** option.

```
prompt> place_eco_cells -cells $cells -use_virtual_connection
prompt> place_eco_cells -cells $cells -use_virtual_connection -no_legalize
```

The following examples use the **-use_virtual_connection** option with **-channel_aware**.

```
prompt> place_eco_cells -cells $cells -use_virtual_connection -channel_aware
prompt> place_eco_cells -cells $cells -use_virtual_connection -no_legalize -channel_aware
```

The following examples show how to use the **-max_fanout** option.

```
prompt> place_eco_cells -unplaced_cells -no_legalize -max_fanout 500

prompt> place_eco_cells -unplaced_cells -max_fanout 500
```

The following examples show how to use the **-ignore_pin_connection** option.

```
prompt> place_eco_cells -unplaced_cells -no_legalize -ignore_pin_connection {CP}

prompt> place_eco_cells -unplaced_cells -ignore_pin_connection {CP A}
```

The following examples show how to use the **-remove_filler_references** option.

```
prompt> set libCells [get_lib_cells {lib1/FILL1 lib1/FILL2}]
prompt> place_eco_cells $ecoCells -remove_filler_references $libCells
```

The following example shows how to use the **-remove_filler_references** option in minimum_physical_impact legalization mode.

```
prompt> place_eco_cells $ecoCells -legalize_mode minimum_physical_impact \
   -remove_filler_references {FILL1 FILL2}
```

SEE ALSO

add_buffer(2)

add_buffer_on_route(2)
change_link(2)
create_cell(2)
eco_netlist(2)
legalize_placement(2)
size_cell(2)
set_eco_placement_net_weight(2)
report_eco_placement_net_weight(2)
create_virtual_connection(2)
remove_virtual_connections(2)
get_virtual_connections(2)
add_pins_to_virtual_connection(2)
remove_pins_from_virtual_connection(2)
get_attribute(2)
set_attribute(2)

place_freeze_silicon

Places new engineering change order (ECO) cells by swapping out the spare cells. This command is typically used in the freeze silicon ECO flow.

SYNTAX

status **place_freeze_silicon**
[-cells *cell_objects*]
[-lib_cells_for_filler_recovery *lib_cells*]
[-no_spare_cell_swapping]
[-map_spare_cells_only]
[-write_map_file *map_file_name*]
[-min_filler_distance distance]

Data Types

cell_objects collection
lib_cells collection
map_file_name string
distance distance

ARGUMENTS

-cells *cell_objects*

Specifies the ECO cells on which to do spare cell mapping. The given cells must have the is_fs_eco_add attribute as true and the eco_change_status attribute as one of the following:

- create_cell
- add_buffer
- add_buffer_on_route
- change_link
- size_cell

If this option is not used, the command does spare cell mapping for cells with the same is_fs_eco_add and eco_change_status attribute settings as the ECO cells.

-lib_cells_for_filler_recovery *lib_cells*

Specifies the programmable spare cell (PSC) filler library cells to recover unused space of fillers. When ECO cells are mapped to PSC fillers, there may be some unused white space between ECO cells, and these space need to be recovered as new PSC fillers. The command will only use the specified library cells to create new PSC fillers. If this option is not specified, the command

will use all available PSC filler library cells in the reference libraries.

-no_spare_cell_swapping

Move eco cells to targeted spare cell locations without swapping spare cell (called trial map). This option is used to preview the mapping result and do analysis and remap eco cells if needed. The tool first runs a fast placement of the eco cells and then places each eco cell at its closest matching spare cell but doesn not do spare cell swapping. For any trial mapped cell pair (eco cell, spare cell), they are placed at the same location and the attribute **fs_mapped_cell_name** is used to record the mapping relationship. For eco cell, the attribute value is the trial mapped spare cell name and vice versa. The option can be used with the option **-cells**, **-map_spare_cells_only** and **-write_map_file**. This option also supports programmable spare cells.

-map_spare_cells_only

Perform spare cell mapping based on existing eco cell locations without fast placement for the eco cells. The tool just places each eco cell at its closest matching spare cell and swaps out spare cells. The option is used when you want to use the spare cells in the neighborhood of the given eco cells to do swapping. The option can be used after you apply -no_spare_cell_swapping and confirm that the eco cell locations are acceptable, or after you manually adjust the eco cell locations. The option can be used with the option **-cells** and **-no_spare_cell_swapping**. The option cannot be used with **-write_map_file** unless **-no_spare_cell_swapping** is also turned on. This option also supports programmable spare cells.

-write_map_file ***map_file_name***

Writes out eco cell and spare cell mapping file, which is used with **map_freeze_silicon** command. This option alone will not trigger eco cell placement or spare cell mapping, and just writes out the mapping pairs (eco cell, spare cell) according to the attribute **fs_mapped_cell_name**. The command writes the mapping pair to the mapping file only when the eco cell and spare cell have the same location. If you manually move the eco cell after **-no_spare_cell_swapping**, the command disables the mapping pair and omits the pair from the mapping file. The option can be used with the option **-cells** and **-no_spare_cell_swapping**. The option cannot be used with **-map_spare_cells_only** unless **-no_spare_cell_swapping** is also turned on. This option also supports programmable spare cells.

-min_filler_distance ***distance***

Specify the minimum distance between eco cells after mapping to PSC fillers. The unit of distance is micron, and this distance will be adjusted internally to be an integral multiple of the unit width of PSC fillers. If not specified, the default value is 0 and means no limitation on the spacing. This option only support eco cells with PSC type.

DESCRIPTION

The **place_freeze_silicon** command automatically places cell instances that were added during an ECO operation. The command does not move cells that are placed previously. Only new cells added by an ECO are placed.

By default, the tool maps new ECO cells with existing spare cells of the same reference library cell or maps them with existing programmable spare cells of the same PSC type.

The tool first runs a fast coarse placement for the ECO cells. Next, the tool finds spare cells nearby the ECO cells and then do matching between ECO cells and spare cells. The matching engine will match each ECO cell to its target spare cell as close as possible. Finally, the tool places each ECO cell to the location of its matched spare cell and swaps out the spare cell, and these cells swapped out will be deleted from database.

You can use an ECO routing command to complete the routing. The freeze silicon capability can be used to implement metal-only ECO changes on an existing layout.

If there are not enough spare cells, the command issues error message.

If there are programmable spare cell mapping rules set by command **set_programmable_spare_cell_mapping_rule**, the **place_freeze_silicon** will also honor these mapping rules and execute the specified mapping between eco cells and

programmable spare cells. User can review the current mapping rules by using command **report_programmable_spare_cell_mapping_rule**. The mapping rules also can be removed by using command **remove_programmable_spare_cell_mapping_rule**. If you want that **place_freeze_silicon** does not honor any mapping rule, please remove all the mapping rules first.

The command behavior is the same, regardless of the design.eco_freeze_silicon_mode application option setting.

The option **-no_spare_cell_swapping** and **-map_spare_cells_only** allow user to analyze QoR before committing the spare cell mapping. And provide flexibility for user to manage spare cell mapping for the desired QoR. The user-driven freeze silicon QoR tuning flow is as follows. Additional iterations might be needed between step 3 and step 4.

1. Apply ECO changes.

2. Analyze eco cells and spare cells by **check_freeze_silicon** command.

3. Trial map eco cells to spare cells.

4. Do analysis and remap eco cells.

5. Map eco cells to spare cells.

6. PG and ECO Route.

For routability, you can mark spare cells in congested area as fs_dont_use before **place_freeze_silicon**, which prevents these spare cells from being used and might avoid a routing detour.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example maps the user-specified ECO cells, specified with the **-cells** option, to the near spare cells of the same reference library cell. Finally, the command swaps out the spare cells.

```
prompt> place_freeze_silicon \
  -cells [get_cells *eco_cell*]
```

The following example automatically maps all the newly added ECO cells to the near spare cells of the same reference library cell or the programmable spare cells (PSC) of the same PSC type. Finally, the tool swaps out the spare cells. The unused space from the PSC cells will be recovered by the same PSC-type library cells specified by **-cells** option.

```
prompt> place_freeze_silicon \
     -lib_cells_for_filler_recovery {FILL1 FILL2}
```

The following 3 examples shows the usage of trial map eco cells to spare cells. You can preview the mapping result with **-no_spare_cell_swapping** and do analysis.

```
prompt> place_freeze_silicon -no_spare_cell_swapping \
  -write_map_file my_map_file -cells $sorted_eco_cells
prompt> place_freeze_silicon -no_spare_cell_swapping \
  -cells $sorted_eco_cells
prompt> place_freeze_silicon -no_spare_cell_swapping
```

The following 2 examples shows the usage of mapping eco cells to spare cells.

Manual spare cell mapping:

```
prompt> place_freeze_silicon -write_map_file my_map_file
prompt> map_freeze_silicon -map_file my_map_file

Batch mode mapping:
prompt> place_freeze_silicon -map_spare_cells_only
```

The following 2 examples shows the usage of **-no_spare_cell_swapping** and **-map_spare_cells_only**. When the option **-map_spare_cells_only** is used with **-no_spare_cell_swapping**, the **place_freeze_silicon** command will not run fast placement and not do spare cell swapping, just place each eco cell at its closest matching spare cell.

```
prompt> place_freeze_silicon -cells $sorted_eco_cells \
  -no_spare_cell_swapping -map_spare_cells_only
prompt> place_freeze_silicon -cells $sorted_eco_cells \
  -no_spare_cell_swapping -map_spare_cells_only -write_map_file my_map_file
```

The following example shows analysis and remap eco cells.

```
prompt> place_freeze_silicon -no_spare_cell_swapping \
  -cells $sorted_eco_cells
```

Overlay congestion map and spare cells on GUI. Mark spare cells in congested area fs_dont_use.

```
prompt> set sC [get_selection]
prompt> set sp [filter_collection $sC "is_spare_cell == true" ]
prompt> set_attribute $sp fs_dont_use true
prompt> place_freeze_silicon -no_spare_cell_swapping -cells $sorted_eco_cells
```

Evaluate the new trial mapping result and swap spare cell if the result is better.

```
prompt> place_freeze_silicon -map_spare_cells_only
```

Another example:

```
prompt> place_freeze_silicon -cells $sorted_eco_cells -no_spare_cell_swapping
```

Move eco cells to more feasible locations based on visual analysis.

```
prompt> place_freeze_silicon -cells $modified_eco_cells -map_spare_cells_only
```

The following example automatically maps the specified ECO cells with PSC type to the near programmable spare cells (PSC) of the same PSC type. And the size of space between the mapped eco cells will be larger than the specified distance 0.5.

```
prompt> place_freeze_silicon -cells $psc_eco_cells -min_filler_distance 0.5
```

SEE ALSO

check_freeze_silicon(2)
map_freeze_silicon(2)
set_attribute(2)
add_buffer(2)
eco_netlist(2)
legalize_placement(2)
place_eco_cells(2)
route_eco(2)
spread_spare_cells(2)
set_programmable_spare_cell_mapping_rule(2)

report_programmable_spare_cell_mapping_rule(2)
remove_programmable_spare_cell_mapping_rule(2)

place_group_repeaters

Places interconnect repeaters on route.

SYNTAX

status **place_group_repeaters**
-cells *cell_list* |
-repeater_groups *group_id_list*
[-lib_cell_input *pin_name*]
[-lib_cell_output *pin_name*]
[-max_distance_for_incomplete_route *distance*]
[-first_distance *distance*]
[-last_distance *distance*]
[-blockage_aware]
[-horizontal_repeater_spacing *spacing*]
[-vertical_repeater_spacing *spacing*]
[-layer_cutting_distance *layer_scale_list*]

Data Types

cell_list	collection
group_id_list	list
pin_name	string
distance	float
spacing	integer
layer_scale_list	list

ARGUMENTS

-cells *cell_list*

Specifies list of repeaters to be placed in the repeater paths. The two options -cells and -repeater_groups are mutually exclusive, and one of them is required.

-repeater_groups *group_id_list*

Specifies list of repeater group ids. The two options -cells and -repeater_groups are mutually exclusive, and one of them is required.

-lib_cell_input *pin_name*

Specifies pin name when the lib cell has multiple input pins. The default is D.

-lib_cell_output *pin_name*

Specifies pin name when the lib cell has multiple output pins. The default is Q.

-max_distance_for_incomplete_route *distance*

Specifies the maximum gap or distance from driver pin to route, or from load to route. The input value should be greater than 0.

-first_distance *distance*

Specifies the distance between the driver and the first repeater. The default is 0.

-last_distance *distance*

Specifies the distance between the load and the last repeater. The default is 0.

-blockage_aware

Specified with distance-based options. Cells are grouped and assigned cutlines automatically. Computed cell location does not overlap with placement blockages, macros, or lower layer routes.

-horizontal_repeater_spacing *spacing*

Specifies the horizontal spacing between two adjacent repeaters in the horizontal direction in terms of number of site width. For example, if the width of repeater is 9 cell site, then the minimum horizontal spacing is 9. If you specify it to be less than 9, the tool automatically extends the spacing to 9. In this example, to achieve the checkerboard pattern, the horizontal spacing should be 9,13,17,21....

-vertical_repeater_spacing *spacing*

Specifies the vertical spacing between repeaters in the vertical direction in terms of number of site height. When cell is single site height, the minimum vertical spacing is 1 site height. In this example, to achieve a checkerboard pattern, the vertical spacing should be 1,5,9.... When you specify vertical spacing as 4, the tool automatically rounds it to 5. For a double site height library cell, to achieve a checkerboard pattern, you can specify 2,6,10....

-layer_cutting_distance *layer_scale_list*

Specifies the layer cutting distance list. Using this option, you can define input routes and output routes to cell center distances. Specify the option by using the following format: {{layer1 cutting_distance_in_1 cutting_distance_out_1} {layer2 cutting_distance_in_2 cutting_distance_out_2} ...}

DESCRIPTION

This command performs on route placement of interconnect repeaters.

- Support rule-based repeater placement with distance (two pin supernets) and cutline (multiple fanout supernets).
- Smart self-organized (checkerboard) repeater placement based on your intent.
- Cut routes and assign routes to associated repeater output net after repeaters are placed.
- Generic repeater support (not limited to register, buffer, or inverters).

For place_group_repeater -cells, the command applies distance-based repeater placement.

For place_group_repeater -repeater_groups, the command applies cutline-based repeater placement. Before running the place_group_repeater command, define the repeater groups, register path connections (virtual connection) for each set of repeater groups, and cutline for each repeater group by using the set_repeater_group command.

To define register path connections (virtual connection), specify the driver group, path drivers, or path loads for each repeater group, and specify the cells in sequence. The cell connection between groups depends on the cell order within the group.

For example, cell[0] in group 1 and cell[0] in its driver group 2 belong to one register path.

- For the first group, specify -path_drivers;
- For the middle groups, specify -driver_group_id;
- For the last group, specify -driver_group_id and -path_loads;

For -repeater_groups, this command depends on group virtual connection to build cell connectivity, by default. Each input group should have driver group (or path driver) and load group (or path load); otherwise, the command generates an error. If you need to depend on real connection between cells, you need to set the application option, eco.placement.eco_enable_virtual_connection, to false.

EXAMPLES

The following examples show the usages of command.

The following examples show the usages of -cells.

```
prompt> place_group_repeaters -cells $ecoCells -lib_cell_input D
-lib_cell_output Q -max_distance_for_incomplete_route 3.0
prompt> place_group_repeaters -cells $ecoCells -lib_cell_input D -lib_cell_output
Q -first_distance 20 -last_distance 50 -blockage_aware
```

The following examples show the usages of -repeater_groups based on real connection.

```
prompt> set_repeater_group -group_id 1 -cells [get_cells {eco_cell_3 eco_cell_7}]
-cutline {{720 366} {720 386}}
prompt> set_repeater_group -group_id 2 -cells [get_cells {eco_cell_2 eco_cell_6}]
-cutline {{920 366} {920 386}}
prompt> set_repeater_group -group_id 3 -cells [get_cells {eco_cell_1 eco_cell_5}]
-cutline {{1230 330} {1230 350}}
prompt> set_repeater_group -group_id 4 -cells [get_cells {eco_cell_0 eco_cell_4}]
-cutline {{1430 330} {1430 350}}
prompt> set_app_options -name eco.placement.eco_enable_virtual_connection -value false
prompt> place_group_repeaters -repeater_groups {1 2 3 4} -lib_cell_input D
-lib_cell_output Q
```

The following examples show the usages of -repeater_groups based on user-defined virtual connection.

```
prompt> set_repeater_group -group_id 1 -cells $buf_group_1 -cutline {{250  90} {250 105}}
```

```
    -path_drivers $driver_pins
prompt> set_repeater_group -group_id 2 -cells $buf_group_2 -cutline {{190 130} {205 130}}
-driver_group_id 1
prompt> set_repeater_group -group_id 3 -cells $buf_group_3 -cutline {{165 155} {165 175}}
-driver_group_id 2
prompt> set_repeater_group -group_id 4 -cells $buf_group_4 -cutline {{110 155} {110 175}}
-driver_group_id 3 -path_loads $load_pins_group_4
prompt> set_repeater_group -group_id 5 -cells $buf_group_5 -cutline {{135 220} {150 220}}
-driver_group_id 3 -path_loads $load_pins_group_5
prompt> place_group_repeaters -repeater_groups {1 2 3 4 5}
```

SEE ALSO

set_repeater_group(2)
report_repeater_groups(2)

place_io

Places I/O driver cell instances.

SYNTAX

status **place_io**
[-io_guide *io_guide_list*]
[-rule *rule_name*]
[-matching_types *matching_type_list*]
[-pad_assignment_file *filename*]
[-include_unassigned_pads]
[-incremental]
[-skip_bump_assignment]
[-bump_assignment_only]
[-match_terminals_to_bumps]

Data Types

io_guide_list list
rule_name string
matching_type_list list
filename string

ARGUMENTS

-io_guide *io_guide_list*

Specifies the I/O guides in which to place I/O driver cell instances. Only drivers assigned to the specified I/O guides will be placed. Other drivers will be ignored and remain in their original locations. If no driver cells are assigned to the specified I/O guides, the command will do nothing. If this option is not specified, the command will place I/O drivers in all I/O guides as well as drivers not assigned to any I/O guide.

-rule *rule_name*

Specifies a single routing spacing rule for pathlines. If no rule is specified, the **place_io** command retrieves or creates a rule that satisfies the rules set for all RDL nets. If no rule is set on any RDL nets, the spacing rule is derived from the technology file. This option is used only for flip-chip designs.

-matching_types *matching_type_list*

Specifies the matching types used during I/O placement. If this option is specified, only I/O driver cell instances specified in the matching types will be placed. Other drivers will be ignored and remain in their original locations. If this option is not specified, the command uses all existing matching types and places drivers not part of any matching type as well.

This option is only used for flip-chip designs since matching types describe relationship between bumps and pad pins. There

exist two special types of matching types. The first type contains only bumps. These bumps are excluded from bump assignment and are not connected to any pad pins unless they have been connected to bumps in the original netlist by 2-pin nets. The second type of matching types does not contain any bump. The pads in such type of matching types will excluded from I/O placement. They remain at their current position and are not processed during bump assignment. Their RDL pins will not be connected to any bumps unless they have been connected to bumps in the original netlist by 2-pin nets. If a matching type contains both bumps and pad pins, the **place_io** command assigns bumps to pad pins according to the uniquify number. Excessive bumps or pad pins will remain unassigned.

When this option is selected, logical connectivity of pads that are not part of any specified matching types is not changed unless such pads are connected to bumps contained in the specified matching types. Under such a situation, the pads are disconnected from the bumps.

The creation of matching types enables bump assignment for macros whose design_type attributes are set as macro, not as pad or flip_chip_driver. Normally, because macro cell instances are not considered as I/O cells, no bumps are assigned to their RDL pins. To perform bump assignment for a macro, you must create a matching type that contains the RDL macro pins and bumps to be assigned to these pins. With such a matching type created, the **place_io** command assigns the bumps in the matching type to the pins in the matching type, with the ratio defined by the matching type uniquify number. User can also simply include the macro instance in the matching type. It is equivalent to include all macro RDL pins in the matching type. If user wants the bump to be connected to a specific macro pin terminal, the terminal must be included in the matching type.

When a pin has multiple terminals, only one is connected to the bump assigned. To force the tool to connect all, you must create a matching type with all terminals and the bump. In addition, the uniquify number of the matching type need to be set to the ratio of terminals to bumps.

-pad_assignment_file ***filename***

Specifies allowable I/O guides for one or more I/O pads. Without such constraints, pads not assigned to I/O guides are placed within an I/O guide determined by the **place_io** command. You can limit the I/O guide candidates to a subset of all I/O guides.

The format of the constraint file is as follows. Each line contains one or more pad names followed by a list of I/O guide names. I/O guides not in the list cannot be used to place the pads in the line. If more than one pad is specified, they must be placed in a pair of curly braces. No curly braces are needed for I/O guides. Each line must end with a semicolon.

```
{pad1 pad2} guide1 guide2;
pad3 guide1 guide3 guide0;
```

-include_unassigned_pads

Assigns any unassigned or unconstrained I/O driver cells to an I/O guide and places the cells. If this option is not specified, the command does not place the unassigned or unconstrained drivers, unless you omit both the **-io_guide** and **-matching_types** options.

-incremental

Places I/O driver cells according to their spacing and offset attributes, only for nonflip-chip designs with no I/O constraints. The **place_io** command does not consider I/O guide dimension or bump cell distribution. I/O constraints are ignored with this option.

-skip_bump_assignment

If this option is used, the tool places IO pads only without performing bump assignment. This option and option "-bump_assignment_only" are mutually exclusive.

-bump_assignment_only

If this option is used, the tool does only bump assignment. No pad placement is done and I/O driver cells are not moved. The tool assigns bumps to I/O driver cells based on current locations of I/O driver cells. All the signal I/O constraints and power I/O constraints will be ignored in this mode. This option does not work with **-pad_assignment_file** option. This option and option "-skip_bump_assignment" are mutually exclusive.

-match_terminals_to_bumps

Matches each terminal of a pin to a bump cell. This option only works for pin terminals that fit the following requirements:

a. The label of the pin terminal must be "bump".

b. There is no one-to-one logical connection between the pin and a bump.

If a pin has multiple terminals fitting these requirements, the tool assigns multiple bumps to the pin. For a pin that does not meet these requirements, the tool assigns bumps to the pin in the default way.

By default, the tool picks only one terminal of a pin that has the "bump" label and assigns one bump to the pin during I/O placement.

DESCRIPTION

This command places I/O driver cell instances in I/O guides. By default, I/O driver cell instances that are not assigned to an I/O guide are automatically assigned to available I/O guides and placed. All bumps are considered.

If any pad has a fixed status, it is not moved. In addition, you can also set signal I/O constraints to specify pad orders, pad locations, and space between pads. It is your responsibility not to introduce any constraint conflicts. For example, if the space between two fixed pads is described in a signal I/O constraint, the value must be equal to the actual space between the two fixed pads.

The orientation of a pad is determined by the orientation of its reference cell and the I/O guide which the pad is placed in. Specifically, if the pad is placed in an I/O guide with side set to bottom, its orientation is the same as that of the reference cell. If the pad is placed in I/O guides with different side values, the pad will be rotated in the same way as that of I/O guide with respect to a bottom guide. You can change the reference cell orientation by setting the reference_orientation attribute of the reference cell. Consequently, pad orientations will be changed accordingly. For example, if users set the reference_orientation to MY, a pad placed in a bottom guide will be y-axis flipped. The following command sets the reference_orientation to R270 for all pads whose ref_block attribute is the same as that of **pad1**. As a result, those pads will be rotated by 270 degrees clockwise.

This command ignores non-IO cells. This may cause an overlap between non-IO cells and IO pads. In case, such an overlap needs to be avoided the user can create matching types including all and only such non IO cells. The command will then treat all such non IO cells as regular IO pads and fix their position.

This command will delete all the unfixed fillers in the design.

```
prompt> set_attribute -name reference_orientation \
  [ get_attribute -name ref_block [get_cells pad1]] -value R270
```

You can flip a pad along the direction perpendicular to the I/O guide of the pad. Specifically, you can set the is_flipped attribute as follows. This attribute can only be set on I/O cells placed during I/O placement.

```
prompt> set_attribute -name is_flipped -value true PAD1
```

EXAMPLES

The following example places I/O drivers.

```
prompt> place_io
```

SEE ALSO

create_io_guide(2)
create_matching_type(2)
set_power_io_constraints(2)
set_signal_io_constraints(2)
write_io_constraints(2)

place_opt

Place and optimize the current design.

SYNTAX

place_opt
[-list_only]
[-from *startStage*]
[-to *endStage*]

Data Types

startStage string
endStage string

ARGUMENTS

-list_only

Prints out the major stages that constitute the default flow for the **place_opt** command.

These stages are: **initial_place**, **initial_drc**, **initial_opto**, **final_place**, and **final_opto**.

This option cannot be used with the **-from** or **-to** options.

-from *startStage*

Specifies the starting stage for the **place_opt** command. Valid values are **initial_place**, **initial_drc**, **initial_opto**, **final_place**, and **final_opto**.

The default is **initial_place**.

If you specify both the **-to** and **-from** options, the *startStage* stage cannot be preceded by the *endStage* stage in the order specified by the **-list_only** option.

-to *endStage*

Specifies the final stage for the **place_opt** command. Valid values are **initial_place**, **initial_drc**, **initial_opto**, **final_place**, and **final_opto**.

The default is **final_opto**.

If you specify both the **-to** and **-from** options, the *startStage* stage cannot be preceded by the *endStage* stage in the order specified by the **-list_only** option.

DESCRIPTION

The default behavior of this command (1) places the current design, (2) optimizes the placed design for timing, electrical DRC violations, area, power, and routability, (3) performs incremental placement to optimize timing and routability, (4) further optimizes the placed design, and (5) legalizes the design placement at the end.

EXAMPLES

The following example places and optimizes the current design and ends up with an optimized design with a legalized placement:

```
prompt> place_opt
```

The following example takes in a placed design as its input and optimizes it for timing, electrical DRC violations, power, area, and routability, without rerunning placement on it, and ends up with an optimized design with a legalized placement:

```
prompt> place_opt -from final_opto
```

SEE ALSO

clock_opt(2)
refine_opt(2)

place_pins

Places pins on blocks within the current design or on the current block.

SYNTAX

status **place_pins**
[-cells *cells*]
[-nets *nets*]
[-pins *pins*]
[-ports *ports*]
[-exclude_nets *nets*]
[-nets_to_exclude_from_routing *nets*]
[-self]
[-legalize]
[-use_existing_routing]

Data Types

cells	collection
nets	collection
pins	collection
ports	collection

ARGUMENTS

-cells *cells*

Specifies the blocks for pin assignment. Only the nets connected to the specified cells are routed during global routing and only pins on selected blocks are placed. By default, the command performs pin assignment on all blocks.

-nets *nets*

Specifies the nets for pin assignment. Only the specified nets are routed during global routing stage and only block pins connected to the selected nets are placed. By default, the command performs pin assignment on all nets.

-pins *pins*

Specifies the pins for pin assignment. Only the specified pins are placed when this option is used. The tool routes the nets that are connected to the specified pins but the tool skips feedthrough creation on those nets. So no feedthrough nets or pins are created or placed. By default, the command performs pin assignment on all block pins.

-ports *ports*

Specifies the top-level ports for pin assignment. The command places only the terminals associated with the specified ports. By default, the tool performs pin assignment on all block pins.

-exclude_nets *nets*

Specifies the nets that are excluded for pin assignment and feedthrough creation. All nets, except the excluded nets, are routed and pins are placed based on the routed nets. All the pins including feedthrough pins are placed except the pins that connected to those specified nets.

-nets_to_exclude_from_routing *nets*

Specifies the nets that are excluded from routing during pin placement. By default, the tool tries to route all the nets that are used for pin placement. This option can be useful if some nets are preferred to skip routing to get quick pin placement results on them. When this option is specified, the **place_pins** command performs pin placement based on net connectivity to minimize wire length. This option cannot be used with the option **-use_existing_routing** , since the routing is skipped when using that particular option.

-self

Performs pin placement on the top-level design. By default, pin placement is performed on blocks within the current top-level design.

-legalize

Performs only pin legalization. The command runs in incremental mode (the value of app option **plan.pins.incremental** is ignored) and legalizes pin overlaps, pin shorts with pre-routes and off-track pins. Routing of nets is skipped, this can be used to quickly legalize existing pin placement. With this option pins that are already legal are not moved and pin shapes are not created for this pins/ports that do not already have shapes associated. That is only existing pins are legalized if needed. Existing pin shapes are considered legal if they are on edge, on track, do not have any overlaps and do not violate any technology related spacing constraints (like layer minimum spacing). Note that the app option to synthesize abutted pins is ignored during the legalize mode of the command.

This option cannot be used with the **-use_existing_routing** or the **-nets_to_exclude_from_routing** options.

-use_existing_routing

Uses existing global routing when placing pins. When this option is specified, the command **place_pins** expects that the nets of the relevant pins are routed by proper global routing commands already, for example, **route_global -virtual_flat** or **route_group -global_planning true**. These routing results will be removed after **place_pins** completes.

When this option is specified and no routing exists for a net, the locations of the relevant pins might not be optimal. The command **place_pins** will not try to route those nets. It will not calculate the flyline based location either.

Note that when running global route before **place_pins -use_existing_routing**, if the block pins already exist in the design, then the global router will route to those pre-existing pins. As a result, if pins are then placed by using these existing routing, their locations might have little or no changes.

Note that this option cannot be used with the **-pins** option.

DESCRIPTION

This command places pins on blocks within the current design, or the top ports of the current design, based on the command line options specified by user. If user does not specify the types of pins to be placed, it places pins for all the editable blocks without fixed physical pins within the current design.

The command first performs global routing, then places pins in legal locations on wire tracks based on the global routing results. The **place_pins** command does not use reserved tracks for pin placement. Depending on the type of pins to be placed, the wire tracks used are:

- For the top ports, **place_pins** uses wire tracks defined at the top.
- For the block pins, **place_pins** uses wire tracks defined inside the corresponding blocks. If the corresponding block does not have any wire tracks defined, then **place_pins** will use the wire tracks defined at the top.

Pin placement also honors pin constraints set by the **create_topological_constraints**, **set_individual_pin_constraints**, **set_bundle_pin_constraints**, and **set_block_pin_constraints** commands. The constraints set by those commands are therefore referred to as topological pin constraints, individual pin constraints, bundle pin constraints, and block pin constraints, respectively. These four types of pin constraints can also be defined in a pin constraints file, which can be read in by command **read_pin_constraints**.

If multiple pin constraints are applied to the same pins, normally the constraints are applied with the following priority:

- topological pin constraints
- individual pin constraints
- bundle pin constraints
- block pin constraints

However, for bundle pin constraints, if the option **-keep_pins_together** is set true, then **place_pins** will try to honor it whenever possible. In some cases, it might have to relax the individual pin constraints requested layer or spacing.

Certain types of pin constraints, e.g. spacing, location, layer, or pin length, can be specified as hard constraints by command **set_block_pin_constraints**. When these constraints are hard, **place_pins** honors the applicable value and skip all DRC rule checks, even if it means possible DRC violations. For example, if location is hard and one pin is constrained at a location inside pin blockage, or will be shorted to another pin, this pin will be placed at the location regardless. Or if length is hard and one pin is constrained by a pin length that is shorter than minimum length, or this value violates min area rule, the pin will still be of the specified length regardless these violations.

A pin guide or a routing corridor can be created to refrain certain pins within the defined region. Alternatively, a pin blockage can be created to forbid certain pins with in the defined region.

If pin constraints are so restrictive that **place_pins** cannot honor all of them, one or more constraints might be relaxed to place the constrained pins, provided they are not hard constraints. Depending on the type of constraint a pin has, the sequence of constraints relaxation can be slightly different.

If the pin has the side and offset constraints, the constraints relaxation sequence is the following:

- If the pin has a non-zero range constraint such as offset range, pin guide, routing corridor, etc, then relax the spacing. Otherwise, relax the offset.
- Relax both offset and spacing.
- Ignore the edge abutment.

If the pin does not have the side and offset constraints, the constraint relaxation sequence is the following:

- Relax spacing.
- Ignore edge abutment but honor spacing.

If the preceding constraints relax sequence cannot find a solution to place the pin, the command tries additional relaxation sequence as follows:

- Relax offset but honor others.
- Relax layer but honor others.
- Relax spacing but honor others.

- Relax offset, layer and spacing.

During constraint relaxation, if any of the constraints are hard constraints, the corresponding relaxation is skipped.

The pin width, length and spacing can be affected by other factors in addition to the pin, bundle or block pin constraints, if they are not specified as hard constraints.

In particular, the pin width is decided by the applicable pin constraints applied to them according to the priority sequence above, with the following exceptions:

- if the pin is placed on a wire track with non-zero track width, then the pin width is the same as the track width. If the track width is less than the default width of the layer, the width is then promoted to the default width. All other width related pin constraints are ignored.
- if the pin has no width-related pin constraints, but its net has a non-default routing rule requiring a width on the layer, then the pin width is same as the non-default rule specified width
- if the pin has no width-related pin constraints and no non-default routing rule, then the pin width is the same as the maximum of the default width and min-width rule defined by the technology file

The pin length is decided as the maximum of the pin length constraints (if applicable/present) and the following:

- the pin width
- the min-length rule defined in the technology file on the layer if it is applicable
- the pin-length required to satisfy the min-area rule if there is a min-area rule defined in the technology file on the layer. Note that width based min-area rules are also supported, the maximum of the min-area rule and width-based min area rule will be applied if both are present.
- lastly, if the layer has the mask constraints defined, the pin length is promoted to 10X routing pitches

Note that the length of the pins after being determined as the maximum of the above mentioned constraints/values, is rounded up to the nearest multiple of the grid size (if grid is present). This is to ensure that pins are created on the manufacturing grid. The depth is rounded to nearest multiple of grid size that is greater than constrained depth.

If there is no individual or bundle pin spacing constraints, then the pin spacing is decided by the following:

- if its net has a non-default routing rule requiring a spacing distance on the layer, then the pin spacing to the nearest pin honors the specified distance
- if its net does not have a non-default routing rule or the non-default routing rule does not have a spacing distance requirement on the layer, then the pin spacing is decided by the command **set_block_pin_constraints**

The **set_editability** command can enable or disable this command for blocks or levels of physical hierarchy. The **place_pins** command has no effect on the blocks that are disabled by **set_editability**.

If feedthrough creation is allowed, this command might create feedthrough nets and ports, depending on the global routing results. If feedthroughs are created, the design netlist is modified. The newly-created feedthroughs are automatically updated with multivoltage (MV) constraints if the design uses UPF.

If feedthroughs pins/nets are created, by default, newly created feedthrough ports and nets are named "net_name__FEEDTHRU_#" in **place_pins**, where net_name is the name of the original net that connected this feedthrough port or feedthrough net, and "#" is an index number. The names of newly-created feedthroughs pins/nets can be controlled by following application options:

plan.pins.new_port_name_tag
plan.pins.new_port_name_tag_style

Use the following application option to control the split-net naming convention:

set_app_options -name plan.pins.new_split_nets_use_block_names -value true

When this application option is true, the tool inserts the feedthrough master block names into the feedthrough net name. The inserted string starts with master name of the driving block, then a sequence of all the load blocks master names separated by double underscore characters. By default, this application option is false by default and block names are not inserted into the feedthrough net name.

For example, if you are using suffix name tag style (default), the feedthrough net name is:

BaseNetName + "__" + BlockBasedSubString + "__" + NameTag + __ IndexNumber

For prefix name tag style:

NameTag + "_" + IndexNumber + "__" + BaseNetName + "__" + BlockBasedSubString

Note that this only applies to split nets that connect one block to one or more other blocks. It DOES NOT apply to any split net that has a top port connection in the block in which the split net belongs.

You can disable pin placement on an a previously-placed pin by setting the physical_status to "fixed" for the terminal of this pin. The following command sets this "fixed" status on terminal of pin block/A and pin placement skips placing this pin.

prompt> **set_attribute [get_terminals -of_objects [get_pins block/A]] physical_status fixed**

By default, pin placement tries to generate the global route and place pins so that block pins can be aligned for nets connecting two pins. To direct the tool to apply more effort to achieve better pin placement, use the **plan.pins.reserved_channel_threshold** application option.

By default, clock nets are excluded from feedthrough creation. To enable feedthrough creation on clock nets, set the **plan.pins.exclude_clocks_from_feedthroughs** application option to false.

When **-self** option is specified, the tool still places pins for the ports that are not marked as power/ground type but connected to power/ground net. When **-self** option is not specified and block pins are connected to top level power/ground net but the block port type are not power/ground, the tool still places pins for those block ports.

Users can use app options to control pin placement behavior. Specifically, pin placement related app options start with plan.pins. The following are current app options related to place_pins. For more details, use report_app_options plan.pins* to list those app options and their current settings.

plan.pins.allow_pins_in_narrow_macro_channels
plan.pins.exclude_clocks_from_feedthroughs
plan.pins.exclude_inout_nets_from_feedthroughs
plan.pins.incremental
plan.pins.layer_range
plan.pins.new_port_name_tag
plan.pins.new_port_name_tag_style
plan.pins.new_split_nets_use_block_names
plan.pins.pin_range
plan.pins.reserved_channel_threshold
plan.pins.retain_bus_name
plan.pins.synthesize_abutted_pins
plan.pins.ignore_pin_spacing_constraint_to_pg
plan.pins.treat_pg_as_shield
plan.pins.enable_span_rules

If there are mask constraints defined on the tracks and if **place_pins** creates any pins on the mask constraints defined tracks, the mask constraints will also be set on the pins as the following:

- The mask constraint set on the pin is from one of the values set on the tracks. **place_pins** will not use any other values not seen on the current tracks.
- **place_pins** first checks the pins nearest usable tracks. A usable track is the track that is not occupied by this pin, and if there is a layer shape on this track with minimum width, it does not trigger the minimum spacing or any mask constraint spacing

violation. The nearest usable tracks are those usable tracks on this pins left and right (or top and bottom) that are nearest.

- Then **place_pins** negates the mask constraints values on those nearest usable tracks. If there are common values between the negated mask constraints values on the left (or bottom) nearest usable track and the right (or top) nearest usable track, **place_pins** picks one value and applied it to this pin.
- If there are no common values between the negated mask constraints on the left (or bottom) nearest usable track and the right (or top) nearest usable track, **place_pins** chooses the mask constraint on the center track that this pin occupies and apply it to this pin. And a warning message shall be issued to the user that no applicable mask constraints values could be found.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following command places pins on all blocks within the current design.

```
prompt> place_pins
```

The following commands enable four cores for pin placement and perform pin placement on all blocks within the current design.

```
prompt> set_host_options -max_cores 4
prompt> place_pins
```

SEE ALSO

route_global(2)
set_block_pin_constraints(2)
set_bundle_pin_constraints(2)
set_editability(2)
set_host_options(2)
set_individual_pin_constraints(2)
write_shadow_eco(2)

pop_up_objects

Transfers physical objects from the block level to the top level.

SYNTAX

string **pop_up_objects**
[-cells *block_instance_collection*]
[-blocks *block_instance_collection*] (OBSOLETED)
object_collection

Data Types

block_instance_collection collection
object_collection collection

ARGUMENTS

-cells *block_instance_collection*

Specifies which block instances from which to pop up objects. By default, all block instances are processed.

object_collection

Specifies the block-level objects to pop up. The objects include nets, routing objects (paths, wires, vias and terminals), routing guides, placement blockages, pin guides, pin blockages and cell instances, or the owner hardmacro cells of the marker layer shapes (see **set_pop_up_object_options** command). For example, if *net_collection* is included with *object_collection*, the *pop_up_objects* command pops up routing objects associated with the specified nets. Note that via matrix is always popped up with its owner net. So if and only if user specify the owner net in the *net_collection*, the via matrix can be popped up.

The *object_collection* option is required for the *pop_up_objects* command.

DESCRIPTION

This command transfers objects from block level to the top level.

If the **-pins_as_terminals** option has been set with the **set_pop_up_object_options** command, and routing for a block net which has a terminal is being popped up, a terminal is created in the top block if the net in the top block has a logical connection to a top-level port. A terminal will be created if the child net has been specified for pop-up, or if the block terminal has been specified for pop-up. If a specific object of a child net has been specified in the collection, but not the entire net, and not the specific block terminal, then no terminal will be created.

If a block-level net is specified for pop-up, then all routing objects will be popped up, unless no paths or vias exist in the block for

that net. If only select shapes of a net are specified for pop-up, then only those shapes will be popped up.

In other words, to pop-up an entire net and remove it from the logical netlist for its current block, you must specify the net and not the shapes of the net. If you specify only the shapes for pop-up, the net itself is not popped up.

If the specified net for pop-up is part of a block that has multiple instances (MIB), then if any of the instances are have not net connection for the port related to the net, then no routing is popped up for that instance.

If routing blockages or placement blockages in a block are the result of pushing down cells or routing as blockages, then these blockages cannot be popped back to the top. The will have the "copied-down" shadow status, and no objects with this shadow status may be popped back to the top.

Attributes are copied from the corresponding block-level object for all objects created at the parent level.

EXAMPLES

The following example pops up the power and ground net routing objects.

```
prompt> set pg_nets [get_nets {blockA/VDD blockA/VSS}]
prompt> pop_up_objects $pg_nets
```

The following example pops up cell instances from block block_1.

```
prompt> set pop_cells [get_cells {blockA/U1 blockA/U2}]
prompt> pop_up_objects $pop_cells
```

The following example pops up signal routing objects (using an alternate way of specifying block-level objects for pop-u->p):

```
prompt> set pop_nets [get_nets blockA/netA]
prompt> pop_up_objects $pop_nets
```

The following example pops up a specific block-level via:

```
prompt> set pop_vias [get_vias blockA/VIA_C_0]
prompt> pop_up_objects $pop_vias
```

SEE ALSO

push_down_objects(2)
remove_pop_up_object_options(2)
report_pop_up_object_options(2)
set_pop_up_object_options(2)

preview

Evaluates its arg list without actually executing it, but option values are "stored through" if option value-tracking is enabled.

SYNTAX

preview *arg* ...

ARGUMENTS

arg

The arg list will be evaluated without being executed.

DESCRIPTION

The preview command (like the built-in Tcl command eval) takes one or more arguments, which together comprise a Tcl script containing one or more commands. The preview command concatenates all its arguments together (like eval), and passes the concatenated string to the CCI/Tcl interpreter recursively.

The preview command passes its concatenated command string to the CCI/Tcl command evaluator as usual, but the special feature of the preview command is that the command will not actually be executed by preview. Instead only the early stages of CCI argument/option checking will happen for the command string. Specifically, standard CCI checking of option arguments is performed, as is "store through" of new current option values for options with current-value-tracking enabled (if option checking succeeded). The preview command thus provides a way to update current values of a command's options (which have current-value-tracking enabled) without actually executing the command.

Conceptually, the CCI preview command is in the same "family" as the Tcl built-in command eval - the difference is that eval executes its concatenated command string, while preview does not, but preview does implement CCI option checking and "store through" for current values of options (which have current-value-tracking enabled).

The advantage of the preview command (compared with a command for turning on and off preview mode) is that a preview command is guaranteed to finish/exit (thus guaranteeing that the preview behavior in the CCI command evaluator will get turned off as soon as the preview command exits). If instead it was necessary to have matching commands for turning on and off "preview mode", bugs in Tcl script code might cause the "turn off preview mode" call to be skipped - then the interpreter would be stuck in "preview mode" indefinitely (and we do not want to take the risk of such a serious bug happening). By having preview as a command, we greatly reduce the risk that the interpreter could get "stuck in preview mode".

Note that the preview command itself does not implement any additional echoing to output of its concatenated command.

About preview and mutually exclusive options: preview performs all option checking for the previewed command as if it was to be executed (even though the command will not be executed). Thus the simultaneous presence of multiple mutually exclusive options

will generate an error under preview (and "store through" of current option values will be aborted). This can be regarded as both a feature (it's consistent with normal command execution) and a limitation (it makes it impossible to set current values for mutually exclusive options of a command from a single run of preview). To workaround this limitation, we also provide the set_command_option_value command (which does not do checks such as the mutual exclusion check) - see the man page for set_command_option_value.

A possible power user application: power users could put preview commands in their home directory Tcl startup files for their favorite dialogs/commands to "seed" initial current values for these dialogs. This may lower the need for automatically saving replay scripts (of preview commands for each command/dialog) on exit from the application. Note: the set_command_option_value command may be easier to use than preview for this purpose.

EXAMPLES

Consider a command named foo with two integer options, -bar1 and -bar2, that have current-value-tracking enabled. In the following example, the "current values" being tracked for the -bar1 and -bar2 options are updated to 10 and 20 respectively:

```
prompt> preview foo  -bar1 10  -bar2 20
prompt> get_command_option_values -current -command foo
-bar1 10  -bar2 20
```

Note that foo command is not actually executed in this example.

SEE ALSO

get_command_option_values(2)
set_command_option_value(2)

preview_dft

Runs DFT Architect Only.

SYNTAX

status **preview_dft**

ARGUMENTS

This command has no arguments.

DESCRIPTION

Previews, but does not implement, scan chains, DFT signals connections and the on-chip clocking control logic to be added to the current design.

EXAMPLES

```
prompt> set_dft_signal -type ScanEnable -port SE
prompt> set_dft_signal -type TestMode -port TM
prompt> set_dft_signal -type ScanDataIn -port si*
prompt> set_dft_signal -type ScanDataOut -port so*
prompt> create_test_protocol
prompt> set_scan_configuration -chain_count 2
prompt> set_scan_compression_configuration -chain_count 20 -input 6 -output 6
prompt> preview_dft
preview_dft
Information: Architecting DFT Scan IP with '2' scan chains
Information: Architecting DFT Compressor IP with '6' inputs / '6' outputs / '20' internal chains
ScanDataIn:
"si0" -> U_DFT_TOP_IP/si[0]
"si1" -> U_DFT_TOP_IP/si[1]
"si2" -> U_DFT_TOP_IP/si[2]
"si3" -> U_DFT_TOP_IP/si[3]
"si4" -> U_DFT_TOP_IP/si[4]
"si5" -> U_DFT_TOP_IP/si[5]
ScanDataOut:
```

"so0" -> U_DFT_TOP_IP/so[0]
"so1" -> U_DFT_TOP_IP/so[1]
"so2" -> U_DFT_TOP_IP/so[2]
"so3" -> U_DFT_TOP_IP/so[3]
"so4" -> U_DFT_TOP_IP/so[4]
"so5" -> U_DFT_TOP_IP/so[5]
TestMode:
"TM" -> U_DFT_TOP_IP/test_mode[0]
ScanEnable:
"SE" -> U_DFT_TOP_IP/test_se
1

SEE ALSO

insert_dft(2)
set_dft_signal(2)
set_scan_configuration(2)
set_scan_compression_configuration(2)

print_message_info

Prints information about diagnostic messages that have occurred or have been limited.

SYNTAX

string **print_message_info**
  [-ids *id_list*]
  [-summary]

Data Types

id_list list

ARGUMENTS

-ids *id_list*

Specifies a list of message identifiers to report. Each entry can be a specific message or a glob-style pattern that matches one or more messages. If this option is omitted and no other options are given, then all messages that have occurred or have been limited are reported.

-summary

Generates a summary of error, warning, and informational messages that have occurred so far.

DESCRIPTION

The **print_message_info** command enables you to print summary information about error, warning, and informational messages that have occurred or have been limited with the **set_message_info** command. For example, if the following message is generated, information about it is recorded:

Error: unknown command 'wrong_command' (CMD-005)

It is useful to be able to summarize all recorded information about generated diagnostic messages. Much of this can be done using the **get_message_info** command, but you need to know a specific message ID. By default, **print_message_info** summarizes all of the information. It provides a single line for each message that has occurred or has been limited, and one summary line that shows the total number of errors, warnings, and informational messages that have occurred so far. If an *id_list* is given, then only messages matching those patterns are displayed. If **-summary** is given, then a summary is displayed.

Using a pattern in the *id_list* is intended to show a specific message prefix, for example, "CMD*". Note that this does not show all messages with that prefix. It shows only the messages that have occurred or have been limited.

EXAMPLES

The following example uses **print_message_info** to show a few specific messages:

```
prompt> print_message_info -ids [list "CMD*" APP-99]

Id          Limit   Occurrences   Suppressed
-------------------------------------------------------------
CMD-005       0          7          2
APP-99        1          0          0
```

At the end of the session, you might want to generate some information about a set of interesting messages, such as how many times each occurred (which includes suppressions), how many times each was suppressed, and whether a limit was set for any of them. The following example uses **print_message_info** to get this information:

```
prompt> print_message_info

Id          Limit   Occurrences   Suppressed
-------------------------------------------------------------
CMD-005       0          12         0
APP-027     100          150        50
APP-99        0          1          0

Diagnostics summary: 12 errors, 150 warnings, 1 informational
```

Note that the suppressed count is not necessarily the difference between the limit and the occurrences, since the limit can be dynamically changed with the **set_message_info** command.

SEE ALSO

get_message_info(2)
set_message_info(2)
suppress_message(2)

print_proc_new_vars

Checks for new variables created within a Tcl procedure.

SYNTAX

string **print_proc_new_vars**

ARGUMENTS

The **print_proc_new_vars** command has no arguments.

DESCRIPTION

The **print_proc_new_vars** command is used in a Tcl procedure to show new variables created up to that point in the procedure. If the **sh_new_variable_message_in_proc** and **sh_new_variable_message** variables are both set to **true**, this command is enabled. If either variable is **false**, the command does nothing. The result of the command is always an empty string.

When enabled, the command issues a CMD-041 message for each variable created in the scope of the procedure so far. Arguments to the procedure are excluded.

For procedures that are in a namespace (that is, not in the global scope), **print_proc_new_vars** also requires that you have defined some aspect of the procedure using the **define_proc_attributes** command.

Important: This feature has a significant negative impact on CPU performance. This should be used only when developing scripts or in interactive use. When you turn on the feature, the application issues a message (CMD-042) to warn you about the usage of this feature.

EXAMPLES

Assuming that the appropriate control variables are set to **true**, the following commands show new variables created within procedures:

```
prompt> proc new_proc {a} {
        set x $a
        set y $x
        unset x
        print_proc_new_vars
```

```
        }

prompt> new_proc 67
=New variable report for new_proc:
Information: Defining new variable 'y'. (CMD-041)
=END=

prompt>

prompt> proc ns::p2 {a} {
          set x $a
          print_proc_new_vars
        }

prompt> define_proc_attributes ns::p2

prompt> ns::p2 67
=New variable report for ns::p2:
Information: Defining new variable 'x'. (CMD-041)
=END=
```

SEE ALSO

define_proc_attributes(2)
sh_new_variable_message(3)
sh_new_variable_message_in_proc(3)

print_suppressed_messages

Displays an alphabetical list of message IDs that are currently suppressed.

SYNTAX

string **print_suppressed_messages**

ARGUMENTS

The **print_suppressed_messages** command has no arguments.

DESCRIPTION

The **print_suppressed_messages** command displays all messages that you suppressed using the **suppress_message** command. The messages are listed in alphabetical order. You only can suppress informational and warning messages. The result of **print_suppressed_messages** is always the empty string.

EXAMPLES

The following example shows the output from the **print_suppressed_messages** command:

```
prompt> print_suppressed_messages
No messages are suppressed

prompt> suppress_message {XYZ-001 CMD-029 UI-1}

prompt> print_suppressed_messages
The following 3 messages are suppressed:
    CMD-029, UI-1, XYZ-001
```

SEE ALSO

suppress_message(2)
unsuppress_message(2)

printenv

Prints the value of environment variables.

SYNTAX

string **printenv**
  [*variable_name*]

Data Types

variable_name string

ARGUMENTS

variable_name

Specifies the name of a single environment variable to print.

DESCRIPTION

The **printenv** command prints the values of the environment variables inherited from the parent process or set from within the application with the **setenv** command. When no arguments are specified, all environment variables are listed. When a single *variable_name* is specified, if that variable exists, its value is printed. There is no useful return value from **printenv**.

To retrieve the value of a single environment variable, use the **getenv** command.

EXAMPLES

The following examples show the output from the **printenv** command:

```
prompt> printenv SHELL
/bin/csh

prompt> printenv EDITOR
emacs
```

SEE ALSO

getenv(2)
setenv(2)

printvar

Prints the values of one or more variables.

SYNTAX

string **printvar**
[*pattern*]
[-user_defined | -application]

Data Types

pattern string

ARGUMENTS

pattern

Prints variable names that match *pattern*. The optional *pattern* argument can include the wildcard characters * (asterisk) and ? (question mark). If not specified, all variables are printed.

-user_defined

Shows only user-defined variables. This option is mutually exclusive with the **-application** option.

-application

Shows only application variables. This option is mutually exclusive with the **-user_defined** option.

DESCRIPTION

The **printvar** command prints the values of one or more variables. If the *pattern* argument is not specified, the command prints out the values of application and user-defined variables. The **-user_defined** option limits the variables to those that you have defined. The **-application** option limits the variables to those created by the application. The two options are mutually exclusive and cannot be used together.

Some variables are suppressed from the output of **printvar**. For example, the Tcl environment variable array *env* is not shown. To see the environment variables, use the **printenv** command. Also, the Tcl variable **errorInfo** is not shown. To see the error stack, use the **error_info** command.

EXAMPLES

The following command prints the values of all of the variables:

```
prompt> printvar
```

The following command prints the values of all application variables that match the pattern *sh*a**:

```
prompt> printvar -application sh*a*
sh_arch              = "sparcOS5"
sh_command_abbrev_mode  = "Anywhere"
sh_enable_page_mode     = "false"
sh_new_variable_message = "false"
sh_new_variable_message_in_proc = "false"
sh_source_uses_search_path = "false"
```

The following command prints the value of the **search_path** variable:

```
prompt> printvar search_path
search_path          = ".  /designs/newcpu/v1.6  /lib/cmos"
```

The following command prints the values of the user-defined variables:

```
prompt> printvar -user_defined
a                = "6"
designDir        = "/u/designs"
```

SEE ALSO

error_info(2)
printenv(2)
setenv(2)

proc_args

Displays the formal parameters of a procedure.

SYNTAX

string **proc_args**
proc_name

Data Types

proc_name string

ARGUMENTS

proc_name

Specifies the name of the procedure.

DESCRIPTION

The **proc_args** command is used to display the names of the formal parameters of a user-defined procedure. This command is essentially a synonym for the Tcl builtin command **info** with the *args* argument.

EXAMPLES

This example shows the output of **proc_args** for a simple procedure:

```
prompt> proc plus {a b} { return [expr $a + $b] }

prompt> proc_args plus
a b

prompt> info args plus
a b

prompt>
```

SEE ALSO

info(2)
proc(2)
proc_body(2)

proc_body

Displays the body of a procedure.

SYNTAX

string **proc_body**
proc_name

Data Types

proc_name string

ARGUMENTS

proc_name

Specifies the name of the procedure.

DESCRIPTION

The **proc_body** command is used to display the body (contents) of a user-defined procedure. This command is essentially a synonym for the Tcl builtin command **info** with the *body* argument.

EXAMPLES

This example shows the output of **proc_body** for a simple procedure:

```
prompt> proc plus {a b} { return [expr $a + $b] }

prompt> proc_body plus
 return [expr $a  + $ b]

prompt> info body plus
 return [expr $a  + $ b]

prompt>
```

SEE ALSO

info(2)
proc(2)
proc_args(2)

promote_clock_data

Loads specific clock data from a lower-level of hierarchy at the top level.

SYNTAX

int **promote_clock_data**
[-cells *block_instance_names*]
[-auto_clock connected | local | all]
[-balance_points]
[-mesh_annotations]
[-port_latency]
[-latency_offset]

Data Types

block_instance_names list

ARGUMENTS

-cells *block_instance_names*

Specifies a list of names of the block instances from which to load specific clock data.

If this option is not specified, the data will be promoted from all block instances in bottom-up order of their respective physical hierarchy level.

-balance_points

When this option is specified, balance points set by the **set_clock_balance_points** command are promoted.

This option is mutually exclusive with **-mesh_annotations**, **-port_latency** and **-latency_offset** options.

-auto_clock connected | local | all

Associates clocks at the block level with clocks at the top level. Before promoting balance points, it is necessary to associate clocks at the block level with clocks at the top level. For example, your top- level clock might be called "SYS_CLK", and the same clock in the block might be called "CLK". You can set the association between block clock and top clock by using the **set_block_to_top_map** command, or by using **promote_clock_data -auto_clock** and specifying the same rule. See the **set_block_to_top_map** man page for more details.

You can select one of three rules for auto-clock mapping:

- **connected** - The "connected" rule associates block-level clocks with top-level clocks under the following conditions: If a block clock has its source at a block port (a boundary clock), it can be associated to a top-level clock that propagates to that port. If a block clock has its source on an internal pin sources (a local clock), the top-level clock source must be declared at the same pin.

- **local** - The "local" rule maps connected clocks. In addition, if a block clock has its source on an internal pin sources (a local clock) and has no associated top-level clock, a new top-level clock will be created with the same waveform as the block-level clock. The new top-level clock will be associated with the local block-level clock.
- **all** - The "all" rule maps clocks as in the "local" rule. In addition, if a block clock has its source at a block port (a boundary clock) and no top-level clock propagates to that port, a new top-level clock will be created with the same waveform as the block-level clock. The new top-level clock will be associated with the local block-level clock. The top-level boundary clock will allow timing analysis to be done, but it is not necessarily correct for signoff. You should consider removing or replacing this clock before signoff.

Note that if two block-level clocks are declared at the same boundary port, the "all" rule cannot resolve both of them. This is because it will not automatically declare two conflicting clocks in the top-level block. You must manually declare two top-level clocks, with the appropriate waveforms. After the top clocks exist, **-auto_clock** mapping can associate them with the block clocks.

-mesh_annotations

When this option is specified, annotations set by the **set_annotated_delay** command and **set_annotated_transition** command are promoted.

This option is mutually exclusive with **-balance_points**, **-port_latency** and **-latency_offset** options.

-port_latency

When this option is specified, latency inside the blocks are promoted and set as balance points on the clock input pins of the block.

This option is mutually exclusive with **-balance_points**, **-mesh_annotations** and **-latency_offset** options.

-latency_offset

When this option is specified, offset latency at clock network sink and clock pins of ICG cells will be promoted.

This option is mutually exclusive with **-balance_points**, **-mesh_annotations** and **-port_latency** options.

DESCRIPTION

The **promote_clock_data** command causes specific clock data from lower levels of physical hierarchy to be applied to the top level. The data include mesh annotations set using **set_annotated_delay** and **set_annotated_transition** commands, CTS balance points set using **set_clock_balance_points** command, latency inside the blocks at the clock input pins and offset latency at clock network sink and clock pins of ICG cells. These promoted data are added to any data that are already applied to the top-level design.

You should consider using this command when:

- The annotations and balance points of a lower level block have changed locally and you want to see these data during top-level analysis.
- You have changed the design view of your lower-level block from "abstract" to "design", and your current data only cover the abstract part of the block.
- You need the latency inside the CCD done blocks to be visible to the top-level.

  The **promote_clock_data** command includes analysis of your block's current configuration, where a configuration is defined by the view type ("design" or "abstract") that you have linked for your block's subblocks. It is possible that your current block configuration is bigger than the configuration of your block at the time balance points and annotations were saved on disk. That is, the balance points and annotations on disk may only cover a portion of the instances and nets in your current block. This will happen when your balance points and annotations on disk cover only an "abstract" view and your current design contains a full "design" view. In this case, **promote_clock_data** detects the problem and prints an error message.

When multiple instances from different levels of physical hierarchy are specified using **-cells** option, the data will be promoted from the block instances in bottom-up order of their respective physical hierarchy level.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following example promotes mesh annotations from block instance "block1".

```
prompt> promote_clock_data -cells block1 -mesh_annotations
```

The following promotes balance points from all physical hierarchies.

```
prompt> promote_clock_data -balance_points
```

SEE ALSO

change_view(2)
link_block(2)
report_clock_balance_points(2)
set_block_to_top_map(2)
synthesize_clock_trees(2)
write_script(2)

propagate_3d_connections

Derives logical connections for bumps in a 3DIC design.

SYNTAX

status **propagate_3d_connections**
[-nets *net_list*]
[-check_only]

Data Types

net_list list

ARGUMENTS

-nets *net_list*

Specifies the nets for logical connection propagation. By default, all top-level nets in the design are propagated.

-check_only

Check only physical and logical connectivity, do not perform propagation. By default, the command propagates logical connections.

DESCRIPTION

This command creates logical connection based on bump-pair physical alignment information with top level nets.

EXAMPLES

The following example derives the logical connection of bumps on the net VDD with I/O port.

prompt> **propagate_3d_connections -nets VDD**

SEE ALSO

create_3d_mirror_bumps(2)
set_cell_location(2)

propagate_3d_matching_types

Propagates matching types of bumps, pins and tsv by tracing nets and bump contacts.

SYNTAX

status **propagate_3d_matching_types**
[-from *source_chip_list*]
[-to *target_chip_list*]
[-matching_types *matching_type_list*]
[-check_only]
[-force]

Data Types

source_chip_list list
target_chip_list list
matching_type_list list

ARGUMENTS

-from *source_chip_list*

Specifies the source chips whose matching types are to be propagated. By default, all chip's matching types will be propagated.

-to *target_chip_list*

Specifies the target chips that the specified matching types will be propagated to. By default, all chips will be propagated to.

-matching_types *matching_type_list*

Specifies the matching types in the source chip to be propagated. By default, all matching types will be propagated.

-check_only

Performs a check for matching type propagation, but do not modify the design. By default, the command propagates the matching types and modifies the design.

-force

Assigns new matching types for all related objects. Related objects are objects in the target chip that have connections to the specified matching types in the source chip.

The **-force** option treats all related objects in the target chip as if they don't have matching types.

DESCRIPTION

This command propagates matching types for bumps, pins and tsv by tracing nets and bump contacts. The matching type specifies which bump cells are to be connected to which I/O driver pins. The **create_matching_type** command defines a matching type constraint, and the **assign_3d_interchip_nets** command uses matching types of cells and terminals to do net assignment. This command propagates matching types of cells, pins and terminals to other pins by tracing nets and contacted bumps. Normally, two bumps that are center-aligned, are treated as "contacted" bumps. Moreover, if you set the application option **plan.3dic.bump_alignment_threshold**, two bumps with centers that are less than this value are considered to be "contacted". Note that the new matching types are assigned to the pins of objects.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, "P1" in chip1 and "SOC/BUMP7" in chip2 have been set matching types "PG" and "SIGNAL". The **-force** option lets "PG" override the "SIGNAL" matching type.

```
prompt> propagate_3d_matching_types -from chip1 -to chip2 -matching_types {PG} -force
Information: propagate matching type { 'PG' } from chip chip1 to target chip chip2.
```

In the following example, matching types "SIGNAL" and "PG" have been set to "SIIP/P1" and "i_bump_1" in chip1, these two matching types will propagate to the pins of "SIIP/P2" and "i_bump_2" in chip2. "default" means no matching type has not been set. Because **-matching_types** is not used, both "PG" and "SIGNAL" matching types will be propagated.

```
prompt> propagate_3d_matching_types -from chip1 -to chip2
Information: propagate matching type { 'SIGNAL' } from chip chip1 to target chip chip2.
Information: propagate matching type { 'PG' } from chip chip1 to target chip chip2.
```

SEE ALSO

assign_3d_interchip_nets(2)
create_matching_type(2)
report_matching_types(2)

propagate_pin_mask_constraint

Propagates the multiple-patterning mask constraint from one pin to its abutted pin on the same layer that does not have the multiple-patterning mask constraint.

SYNTAX

status **propagate_pin_mask_constraint**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command propagates the multiple-patterning mask constraint from one pin to its abutted pin on the same layer that does not have multiple-patterning mask constraint.

The abutted pins can be either two block pins or a block pin and a hard macro pin. If one of the pins is a hard macro pin, it must be the pin that has the multiple-patterning mask constraint.

If neither abutted pin has a multiple-patterning mask constraint, this command does not try to set the constraint on either pin.

The **set_editability** can enable/disable this command for blocks or levels of physical hierarchy. This command has no effect on the blocks that are disabled by **set_editability**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example propagates the multiple-patterning mask constraint on abutted pins.

```
prompt> propagate_pin_mask_constraint
1
```

SEE ALSO

place_pins(2)
check_pin_placement(2)
set_editability(2)

propagate_pin_mask_to_via_metal

SYNTAX

status propagate_pin_mask_to_via_metal
[-block *block*]

Data Types

block	string or collection

ARGUMENTS

-block *block*

Specify the name of the block to be processed.

DESCRIPTION

A via consists of three layers: upper-metal enclosure, via cut, and lower-metal enclosure. When the router drops a via on a pin, the lower-metal enclosure layer of the via is the same as the pin layer.

In a double-patterning technology, each pin-shape instance has an attribute mask_constraint, which identifies the mask to which the shape belongs, such as mask_one or mask_two. Each via instance also has attributes to identify the mask to which each of its three layers belongs.

For stream-out of design data in GDSII format, if a via's lower-metal mask attribute (lower_mask_constraint) is set to no_mask, the tool can automatically propagate the mask identity from an M1 pin-shape to the M1 layer of the via above the pin, overriding the no_mask attribute setting. To do this, use the '-propagate_pin_mask_to_via_metal' option of the write_gds command.

Some applications need the mask identity propagated to the via M1 layers to be reflected in the database. This command can be used for that purpose.

By default, this command operates on the current design. After the command is run, the M1 layer of each via instance is annotated to indicate the same mask information as the M1 pin-shape object under the via. To preserve the modified design data, save the design with the save_block command.

To operate on a design other than the current design, use the -block option.

EXAMPLES

The following command modifies the vias in the current block by updating the lower_mask_constraint with the mask_constraint attribute of the pin-shape on which the via is incident:

```
prompt> propagate_pin_mask_to_via_metal
1
```

The following command modifies the vias in the block 'blockB' by updating the lower_mask_constraint with the mask_constraint attribute of the pin-shape on which the via is incident:

```
prompt> propagate_pin_mask_to_via_metal -block blockB
1
```

SEE ALSO

save_block(2)
write_gds(2)

propagate_switching_activity

Forces the propagation of power-switching activity information.

SYNTAX

status **propagate_switching_activity**
[-scenarios *scenario_list*]
[-modes *mode_list*]
[-corners *corner_list*]

Data Types

scenario_list	list
mode_list	list
corner_list	list

ARGUMENTS

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Activity propagation is done separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, propagation is done for the current scenario in the design.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering on modes will occur.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If this option is not specified, no filtering on corners will occur.

DESCRIPTION

Use this command to force propagation of the power switching activity information.

During the propagation of the switching activity, the information annotated using the read_saif, set_switching_activity commands is used to estimate the activity on unannotated nets. All commands that need switching activity information on all design objects use the switching activity propagation mechanism automatically.

The mechanism to propagate switching activity uses the static probability and toggle rate information you annotate to estimate the static probability and toggle rate information on unannotated design objects. The mechanism used by the tool is based on a stochastic simulation method, which is that random activity based on the annotated switching activity is generated on the annotated design objects, and this activity is propagated across the design using a zero-delay simulator. This is repeated a few thousand times, depending on the effort level specified through power.propagation_effort app option.

Once propagated, activity is retained until a design changes causes the activity to be invalidated. If you wish to be informed of this, you can un-suppress the messages POW-001 and POW-051 with the **unsuppress_message** command. These messages are suppressed by default.

Multicorner-Multimode Support

SEE ALSO

get_switching_activity(2)
report_power(2)
read_saif(2)
set_switching_activity(2)
power.propagation_effort(3)

push_down_clock_trunks

Pushes down top-level clock trunk cells. The cells are pushed into blocks which have been enabled for feedthrough creation. This command is needed only in CTP-Pushdown flavor.

SYNTAX

status **push_down_clock_trunks**
[-clock *clocks*]

Data Types

clocks collection

ARGUMENTS

-clock *clocks*

Specifies a list of clocks for which the clock trunk cells must be pushed down. By default, all top-level clock cells are pushed down.

DESCRIPTION

During clock trunk planning, feedthroughs can be planned by creating clock buffers and physically placing them on the top of a block. Initially, clock buffers created in this way belong to the top level. When planning is complete, the top-level clock buffers must be pushed down into blocks, and feedthroughs must be created. After push down, any remaining unplaced block ports, in particular the newly created ports from the feedthroughs, must be placed.

This command pushes down the clock buffers, creates feedthroughs, and places the newly created block ports. This command is typically used at the end of an automatic or manual clock trunk planning session.

Cells are only pushed down into blocks which have been enabled for feedthrough creation with the **set_block_pin_constraints -allow_feedthroughs** command.

This command is needed only in CTP-Pushdown flavor where blocks are in readonly mode.

EXAMPLES

The following example pushes down clock cells on top of blocks for all clocks.

```
prompt> push_down_clock_trunks
```

SEE ALSO

gui_load_clock_trunk_planning(2)
set_block_pin_constraints(2)

push_down_objects

Pushes top-level objects into blocks.

SYNTAX

string **push_down_objects**
[-cells *block_instance_collection*]
object_collection

Data Types

block_instance_collection collection
object_collection collection

ARGUMENTS

-cells *block_instance_collection*

Specifies which block instances to push down objects into. By default, all blocks are processed.

object_collection

Specifies the object collection to be pushed down. The objects include nets, routing objects (paths, vias and terminals), routing guides, routing corridors, blockages (placement, shaping, routing), cell rows (wire tracks indirectly when rows are specified), and charging stations (non-primary voltage areas and their associated single-rail repeater cells). Note that the **get_shapes** command only returns a collection of paths; the command does not return vias. To include vias, use the **get_vias** command.

For example, if *net_collection* is included with object_collection, this command pushes down routing objects associated with the specified nets. Note that via matrix is always pushed down with its owner net. So if and only if user specify the owner net in the *net_collection*, the via matrix can be pushed down.

The *object_collection* is required for this command.

The following descriptions describe actions that the command performs on the object at the top level and block level. The actions for each object are specified by using the **set_push_down_object_options** command.

The supported objects that can be pushed down into blocks are **pg_routing**, **signal_routing**, **routing_guide**, **routing_corridor**, **blockage** (placement, shaping, routing), **pin_guide**, **pin_blockage**, **site_rows**, and **cells** and **charging_station**. Each object is described below.

- **pg_routing**
  The **pg_routing** objects include paths, vias and terminals that connect to power and ground nets. If a net connects to these objects at the top level, but does not exist in the block, the command creates a child net in the block. If a top-level terminal overlaps with the block and connects to a top-level power and ground net, this command copies the terminal into the block as a block pin with the same shape as the terminal. When terminals are pushed down into blocks as block terminals, they remain in the top cell. They are never removed, regardless of the top_action.

If the net that is being pushed into a block has a connection to a block, then that port determines what net in the block that the created routing is assigned to in the block. For instance, if the net being pushed is VDD0 and connects to block A/VDD, then the routing in the block is created for net VDD in the block.

If the net that is being pushed into a block has a nondefault routing rule (NDR), then that NDR will be created for the new net in the child block.

If the design has MIB instances and PG straps crossing MIB instances, the MIB instances are checked for correct horizontal and vertical alignment of the MIB instances. If misalignment is detected, all PG routing into blocks is suppressed. This can be overriden using the **set_push_down_object_options -object_type pg_routing -ignore_misalignment true**.

The valid top actions for routing objects are *keep* and *remove*, and the valid block actions for routing objects are *copy*, *create_pin_shape*, or both options at the same time. The default top action for *pg_routing* is *remove* and the default block action for *pg_routing* is *create_pin_shape*.

For the routing object type,

If you specify **set_push_down_object_options -top_action keep**, the top-level routing object is untouched after running **push_down_objects**.

If you specify **set_push_down_object_options -top_action remove**, the **push_down_objects** command removes the overlapping section of the top-level route object that overlapped with the block boundary. If paths are collinear with the block boundaries, the paths are not removed from the top.

If you specify **set_push_down_object_options -block_action copy**, (mutually exclusive with **create_blockage**), the **push_down_objects** command copies into the block the overlapping section of the top-level object that overlapped with the block. The newly created block-level routing object is connected to its corresponding block net. A block port and net is created if necessary.

If you specify **set_push_down_object_options -block_action create_blockage**, (mutually exclusive with **copy**), the **push_down_objects** command copies into the block as a routing blockage the overlapping section of the top-level object that overlapped with the block boundary. The newly created block-level routing blockage is not associated with any net, and the netlist is unaffected in both the top and block. The original top-level routing object is retained in the top block without regard to the **-top_action** option setting.

If you specify **set_push_down_object_options -block_action create_pin_shape**, the **push_down_objects** command creates pin shapes where the routing object touches the block boundary. However, if the *object_collection* contains only vias and nothing else, then by specifying **set_push_down_object_options -block_action create_pin_shape**, pins will be created for all vias that touches and within the block boundary. A port is created if necessary. If paths are collinear with the block boundaries, the pin shapes are created collinear for the length of the collinear path. All terminals created in blocks are created with fixed status.

Note that pins are created based only upon either top-level terminal overlap with a block or wire overlap with a block (normally at block boundary crossings). Pins are not created from other net-connected shapes such as rectangles (with or without "macro_pin_connect" attributes). Such shapes do get pushed down into the block, but pins are not created derived from these shapes. Also, such shapes that do get pushed into a block are cut by the amount of overlap with the block, and whatever is left in the top is replaced with rectangles with the overlapped part removed.

In the unusual case where the source of the pin creation from collinear overlap is a wire that looks like it is a horizontal wire with its end overlapping perpendicularly over a block edge, but in fact the wire is technically the other type (horizontal-looking but x endpoint are the same x value, or vertical-looking by y endpoints are the same y value), then the pins that result from this scenario will be created as if the wires really are as they appear, overlapping perpendicularly. The wire is still a collinear wire and will not be modified in the top, but the pins will be cut inside the block like a normal perpendicular overlap produces, and any wires created inside the block from the overlap will be created like normal perpendicular overlap as well.

If you specify **set_push_down_object_options -block_action {copy create_pin_shape}**, the **push_down_objects** command creates a copy of the routing object in the block and creates pin shapes where the routing objects touch the block boundaries.

If you specify **set_push_down_object_options -block_action center_pin_from_wire**, the **push_down_objects** command creates a center pin from the wire shapes (paths) passed into the command using [get_shapes]. This block action must be used with great caution, as it will create a center pin for EVERY path shape passed into the command. You must be sure you use this block_action ONLY with a limited number of explicitly specified path shapes.

If you specify **set_push_down_object_options -routing_overlap_check**, the **push_down_objects** command performs additional checks. If you combine the routing overlap check with **set_push_down_object_options -top_action remove** and **set_push_down_object_options -block_action copy** or **set_push_down_object_options -block_action {copy create_pin_shape}**, the **push_down_objects** command checks for overlaps between objects of the same types (wires against wires, vias against vias). If an overlap exists in the design, the command errors out without pushing down the overlapped object. Since command **push_down_objects** does limited overlap checking, you must use more specific DRC checks to prevent DRC violations and shorts. This checking is only done against existing real physical metal objects like wires and vias. It does not check for routing blockages. This applies to both PG routing and signal routing.

By default, when pins (terminals in blocks) are created after push-down of a signal or of PG routing, they are created as squares. The length of the created pin (the distance from the edge that touches a boundary to the opposite pin edge) is the same as the wire width (for the wire that was used to create the pin). However, the pin length can be extended to a length that is longer than the wire width under either of these two conditions: 1) A minLength rule exists for the layer of the pin (where the minLength is larger than the wire width) or 2) a length determined by the minArea rule, where the calculated length is larger than the wire width or the minLength rule.

If the minLength pin length results in the pin extending further into the block than the routing extends, this introduces the possibility of spacing/short violations with objects in the child block. For this reason, the command attempts to only create the pin from PG strap overlap by starting at the end of the strap where it extends into the block, and measuring outward enough to meet minLength/minArea requirements. This can result in the PG pin extending outside the block a small amount. But the amount that it extends should not exceed the top-level PG routing, so that should not result in pin are that is not already covered by top-level PG routing. The best way to avoid this scenario is to try to make sure that all PG straps overlap a block enough to meet minLength/minArea pin rules. And try to avoid placing objects in the block close enough to the block boundary to have spacing/short violations with the pushed-down PG pin.

For pg_routing, the **set_push_down_object_options** command supports two options to help avoid DRC violations in the block: **-collinear_margin** and **-pin_meet_fatwire_rule**. You must also include the **-object_type pg_routing** option.

The **-collinear_margin** option specifies a distance within which the closest edge of a PG route parallel to (but not overlapping) the block boundary is considered a collinear PG route. A normal collinear PG strap is one that runs parallel to an edge, but overlaps it so that one edge is inside the boundary and the other edge is outside the boundary. Normally, PG straps which run parallel, but do not overlap the boundary, are not treated as collinear straps. Collinear straps are only created in the child block as copies of the original top-level strap, and the original strap is left untouched in the top. With this option, the command will include non-overlapping PG straps and include them as copies in the child block so that their DRC effect can be observed in the block. These will appear outside of the block boundary in the child reference block.

By default, this option is -1, and the command only considers overlapping parallel PG straps as collinear straps.

If the value of the **-collinear_margin** option is 0 (as in microns), only non-overlapping parallel straps that exactly abut a boundary edge will be pushed into the block as a collinear strap. If the value is greater than zero, any PG strap parallel to a block edge will be copied into the block as a collinear strap if the closest edge of the strap is within the distance specified in the option.

Note that this expanded margin for collinear routing only captures wires and vias that fall inside the expanded instance margin. It does not capture non-routing shapes such as rectangles and polygons, because such shapes are not considered collinear.

The **-pin_meet_fatwire_rule** Boolean option is false by default. If this option is true, the command ensures that the size of the PG pin is big enough to trigger fat-wire spacing rules within the block. To do this, the command creates a pin that is as long (in the direction perpendicular to the edge) as it is wide. This length is created inside the boundary if it can safely be done. But if the strap which was the basis for the pin only overlapped the block less than that length, then the pin can protrude outside the boundary.

- **signal_routing**

The **signal_routing** objects include paths, vias and terminals that connect to signal nets. Signal nets specified can be in any physical level of the physical hierarchy. If a net connects to these objects in the block in which the net resides, but does not exist in a block instance that the net overlaps, the command creates a child net in the overlapped block.

If the net that is being pushed into a block does have a connection to a block, then that port will determine what net in the block that the created routing is assigned to in the block. For instance, if the net being pushed is FOO and connects to block A/FOO, then the routing in the block is created for net FOO in the block.

If the net that is being pushed into a block has a nondefault routing rule (NDR), then that NDR will be created for the new net in the child block.

The valid top actions for signal routing objects are *keep* and *remove*, and the valid block actions for routing objects are *copy*, *create_pin_shape*, or both options at the same time. The default top action for *signal_routing* is *remove* and the default block action for *signal_routing* is *create_pin_shape*.

If you specify **set_push_down_object_options -routing_overlap_check**, the **push_down_objects** command performs additional checks. If you combine the routing overlap check with **set_push_down_object_options -top_action remove** and **set_push_down_object_options -block_action copy** or **set_push_down_object_options -block_action {copy create_pin_shape}**, the **push_down_objects** command checks for overlaps between objects of the same types (wires against wires, vias against vias). If an overlap exists in the design, the command errors out without pushing down the overlapped object. Since command **push_down_objects** does limited overlap checking, you must use more specific DRC checks to prevent DRC violations and shorts. This checking is only done against existing real physical metal objects like wires and vias. It does not check for routing blockages. This applies to both PG routing and signal routing.

For the signal routing object type, if you specify **set_push_down_object_options -top_action keep**, the top-level signal routing object is untouched after running **push_down_objects**. No ports are created for the blocks into which the routing is pushed, and the routing objects in the blocks are created without connection to any net. Because no ports are created, no pins are created at the boundary if the *top_action* is *keep*. **set_push_down_object_options -block_action create_pin_shapes** should not be used in combination with **set_push_down_object_options -top_action keep**. Note that multiple push-downs of routing objects with *top_action keep* results in multiple copies of the routing created within the blocks.

If you specify **set_push_down_object_options -top_action remove**, the **push_down_objects** command removes the overlapping section of the top-level signal routing object that overlapped the block boundary. If the paths are collinear with the block boundaries, the net is skipped.

If you specify **set_push_down_object_options -block_action copy**, (mutually exclusive with **create_blockage**), the **push_down_objects** command creates a copy of the part that original top-level object overlapped with the block boundary. In addition, the newly created block-level routing object is connected to its corresponding block net. A block port and net are created if necessary.

If you specify **set_push_down_object_options -block_action create_blockage**, (mutually exclusive with **copy**), the **push_down_objects** command copies into the block as a routing blockage the overlapping section of the top-level object that overlapped with the block boundary. The newly created block-level routing blockage is not associated with any net, and the netlist is unaffected in both the top and block. The original top-level routing object is retained in the top block without regard to the **-top_action** option setting.

If you specify **set_push_down_object_options -block_action create_pin_shape**, the **push_down_objects** command creates pin shapes where the routing object touches the block boundary. A port is created if necessary. If you specify **set_push_down_object_options -block_action {copy create_pin_shape}**, the **push_down_objects** command creates a copy of the routing object in the block and creates pin shapes where the routing objects touch the block boundaries.

The command assesses the routing and determines whether feedthrough ports are a potential necessity. By default, feedthroughs are not created unless the command detects pin constraints for the net that indicate that the command is free to allow feedthroughs on the net. If so, and the routing requires a feedthrough port on the block, the feedthrough ports are created automatically. These feedthrough ports are automatically updated for multi-voltage (MV) boundary conditions if the design has UPF constraints.

In the default configuration for allowing feedthroughs for signal nets (false by default), the command creates an extra

terminal on the block for each secondary and more crossings of the block boundary. These are essentially EEQ (electrically equivalent) pins.

If feedthroughs are allowed, and feedthrough pins are created which result from a terminal overlap with a block, the pin which is derived from that terminal will not have user attributes copied from that terminal, as it normally would if the pin were not a feedthrough pin (a pre-existing logical connection).

To control whether feedthroughs are allowed on a net, set a pin constraint by using the **set_individual_pin_constraints -net netname -allow_feedthroughs true** command. This is absolutely necessary if you intend to push down routing into a block and to create feedthroughs. If you don't allow feedthroughs by using the **set_individual_pin_constraints** command, only one port will be created for a net that crosses a block multiple times for nets with no preexisting logical connections. That port direction will be INOUT, and multiple EQ pins will be created in the block for that port. In the top level, the one port will be assigned to original top-level net that crossed the block. Such a behavior is almost always NOT intended. Ensure that you use the **set_individual_pin_constraints -net netname -allow_feedthroughs true** command if you are pushing down routing into a block that the net is not connected to logically, and you expect a feedthrough (multiple ports) to be created.

By default, if feedthroughs are created and new "split" nets are generated as a result of the original net being broken into pieces, the command composes split net names based on the top-level net name. These split net names can be customized using the **plan.pins.new_port_name_tag** and **plan.pins.new_port_name_tag_style** application options. The plan.pins.new_split_nets_use_block_names application option also controls the naming convention.

Note that if a signal is being pushed down for feedthrough creation, and some routing terminates with connections to leaf cells that are on top of blocks that part of the routing can be pushed into, the command will refuse such a net, because the command requires that such leaf cells be included for push-down in such a scenario.

set_app_options -name plan.pins.new_split_nets_use_block_names -value true

If the **plan.pins.new_split_nets_use_block_names** application option is set to true, the command embeds information about the blocks that are involved in the generated split net into these split net names. The master names of these blocks is embedded into a substring of the new net name. This substring starts with the master name of the driving block, followed by a sequence of all the load blocks master names, separated by double underscore characters. This option is false by default.

The names are composed as follows:

If Suffix name tag style (default):

BaseNetName + "__" + BlockBasedSubString + "__" + NameTag + __ IndexNumber

If Prefix name tag style:

NameTag + "_" + IndexNumber + "__" + BaseNetName + "__" + BlockBasedSubString

Note that this only applies to split nets that connect one block to one or more other blocks. It WILL NOT apply to any split net that has a top port connection in the block in which the split net belongs.

Routing objects which have no net associated with them are pushed down into the block with no net association in the block. Because there is no net association, there is no need to create new ports on the block. Any remaining top-level segments of the routing object continue to have no net association.

The command creates block pins for terminals which overlap a block, even if the net has no routing. All terminals created in blocks are created with fixed status. When terminals are pushed down into blocks as block terminals, they remain in the top cell. They are never removed, regardless of the top_action. If a net has preexisting connections to a block and those preexisting connections have preexisting physical pins, but the routing does not touch those pins, such pins will be deleted unless they have a fixed status.

Attributes are copied from the corresponding top-cell object for all objects created in child blocks.

Wire-stubs are supported in signal routing push-down, but only supports scenarios where the routing crosses a given block instance one time. If it crosses a block, exits, and then recrosses the block (either by looping back or if the block is rectilinear

in such a way that the block gets crossed more than one time), then the tool issues an error message and the net is rejected for push-down. This is because such a routing might require multiple child block nets, but for wire-stubs the tool assumes that all routing in a block belongs to one child net.

By default, when pins (terminals in blocks) are created as a result of push-down of a signal or PG routing, they are created as squares. The length of the created pin (the distance from the edge that touches a boundary to the opposite pin edge) is the same as the wire width for the wire that was used to create the pin. However, the pin length might be extended to a length that is longer than the wire width under either of these two conditions: 1) a minLength rule exists for the layer of the pin (where the minLength is larger than the wire width) or 2) a length determined by the minArea rule where the calculated length is larger than the wire width or the minLength rule.

Abutted Pin Synthesis Support

The command supports an option for validating designs with routing pushed into abutting multiply instantiated blocks (MIBs). After the command pushes down all routing and terminals, the command determines if any newly-created MIB pins abut another block edge. If the MIB pins abut a block edit, the tool examines the relationship of the abutting pin with the adjacent block and creates a corresponding pin on the adjacent block. If there is a previously existing logical connection between the new pin (which is a single pin with no abutted pin initially), the tool creates an abutting physical pin to abut the previously single pin. If no logical connection exists with the adjacent block, then either a new pin with a dangling net will be created in the adjacent block (in the event that the single pin is an output pin), or if the single pin is an input pin, then a tie-low net will be used in the adjacent block and an abutting pin will be created.

If the new pin resulting from the **push_down_objects** command on an MIB block is a feedthrough pin, and it abuts with another newly-created MIB feedthrough pin, then a new net will be created to stitch the two MIB feedthrough pins together.

This feature can be activated by setting the **plan.pins.synthesize_abutted_pins** application option to true.

Note that abutted pin synthesis only supports pin propagation for connections created as a result of feedthrough push-down, which will result in a single physical pin per logical connection. Multiple physical pins for the particular block port are not supported in abutted pin synthesis. Such pins are referred to as EEQ (Electrically Equivalent) pins, and these normally result from pushing down PG routing which cross the block boundary at multiple locations, but which only uses one block port for all the crossings.

- **routing_guide** and **blockage**
  If the **routing_guide** or **blockage** (placement, shaping, routing) objects are included in *object_collection*, and the routing guides or blockages overlap the block boundaries, the overlapping section of the top-level object is pushed down into the block. New routing guides or blockages are created based on the nonoverlapping sections and the tool removes the original top routing guides or blockages.

  The valid actions for routing guides and blockages (placement, shaping, routing) are:

  set_push_down_object_options -top_action keep
  set_push_down_object_options -top_action remove
  set_push_down_object_options -block_action copy

  The default actions for the routing guide and blockage objects are:

  set_push_down_object_options -top_action remove
  set_push_down_object_options -block_action copy

  If **set_push_down_object_options -top_action remove** and **set_push_down_object_options -block_action copy** are specified, the tool pushes the overlapping section of the routing guides or blockages down into the block. The top original routing guides and blockages are removed and new routing guides and blockages are created based on the nonoverlapping sections.

  If **set_push_down_object_options -top_action keep** and **set_push_down_object_options -block_action copy** are specified, the overlapping sections are copied into the block and the original top-level physical objects are untouched.

- **pin_guide** and **pin_blockage**
  If the **pin_guide** or **pin_blockage** objects are included in *object_collection*, and the pin guides or pin blockages overlap the

block boundaries, and the associated pin guide or pin blockages objects have connections to the block, and the block is the parent of the pin guide, the whole pin guide or pin blockage is pushed down into the block. New pin guides or pin blockages are created based on the boundaries of the original top pin guides or pin blockages.

For pin blockages specifically, the blockage will be pushed into the overlapped block if the blockage has NO associations at all. If, the blockage does have associations, but not to the current overlapped block, then it is not pushed into the block. So in effect, NO ASSOCIATIONS is equivalent to association with all blocks.

The valid actions for pin guides and pin blockages are:

set_push_down_object_options -top_action keep
set_push_down_object_options -top_action remove
set_push_down_object_options -block_action copy

The default actions for the pin guide and pin blockage are:

set_push_down_object_options -top_action keep
set_push_down_object_options -block_action copy

If **set_push_down_object_options -top_action remove** and **set_push_down_object_options -block_action copy** are specified, the pin guides or pin blockages are pushed down into the block and the top original pin guide and pin blockage are removed.

If **set_push_down_object_options -top_action keep** and **set_push_down_object_options -block_action copy** are specified, the pin guides or pin blockages are copied into the block and the original top-level physical objects are untouched.

- **site_rows**
  If standalone **site rows** (those not part of a site array) are included in the *object_collection*, those standalone site rows that overlap block boundaries are copied into the block. Along with the site rows, wire tracks are automatically pushed down when site rows are pushed down. All wire tracks over the blocks are pushed down. If there are standalone site rows being pushed into the block, the preexisting site rows are erased and site rows are created.

  If there are wire tracks being pushed into the block, all preexisting wire tracks with the "copied_down" shadow status in the block are erased and new wire tracks are created. Wire tracks are not removed from the top block during this site row and wire track push-down.

  Note that if MIB block instances are present and are misaligned, then this will be detected by the command. If all block instances are considered for push-down (the default... meaning the -cells option is not used) then site rows will not be pushed into any blocks. This can be worked around by either setting the **-ignore_misalignment** option in **set_push_down_object_options** to true:

  Or, you can push down into specific non-MIB instances using the **-cells** option. For non-MIB instances, the use of **-cells** will disregard any MIB misalignment.

  Or just fix the MIB misalignment issue and run site row push-down again.

  The are no top or block actions for site rows.

- **site_arrays**
  If **site arrays** are included in the *object_collection*, site arrays that overlap block boundaries are copied into the block. Along with the site arrays, wire tracks which are non-derived (they don't belong to any site array in the parent block) are automatically pushed down when site arrays are pushed down. All non-derived wire tracks over the blocks are pushed down. If there are preexisting wire tracks in the block, the preexisting non-derived wire tracks with the copied-down shadow status or site arrays of the same name are erased and new wire tracks and site arrays are created. Wire tracks are not removed from the top site during this site array and wire track push-down. The exception to this rule is that if wire tracks exist in the top block that belong to a site array that is pushed down, then some, or all of such site-array-owned tracks will be removed from the top. If a default site array is specified for push-down into a block, any existing default site array (along with any preexisting non-site-array standalone site rows) in the block is deleted before push-down, and replaced with the new pushed-down default site array. Default site arrays are never removed from the top after push-down. Nondefault site arrays

are removed automatically from the top block if the site array boundary is completely enclosed within the block instance boundary, but kept otherwise.

Note that if user-created wire tracks exist in a block that a site array is pushed into, these will not automatically be deleted during wire track push-down associated with site-array push-down. The command will ONLY pre-delete wire tracks which have the "copied_down" shadow status.

There are no top or block actions for site arrays.

- **cells**
  If **cells** are included in the *object_collection*, cell instances that overlap block boundaries are copied into the block that is overlapped. Cells specified for push-down can reside in any physical level of the top-design's physical hierarchy. When multiple cells are pushed down, any nets that connect to only these cells are automatically pushed down into the block also. Any routing associated with these internal nets is automatically pushed down also. The same is true for any cells that have a preexisting connection to the block, and that net does not connect to any other cell besides one (or more) being pushed down. This is a mandatory action because the top-level net is removed from the top-level cell. This action is not controlled by any option which can be set with the **set_push_down_object_options** command.

  If the cells being pushed into the specified block are connected to PG nets, then the command will connect the cells in the block to the child-level PG nets that are connected to the block in the top. If, for some reason, these PG nets are not connected to the block in the top, the command will create new PG ports and nets and connect the cells in the block to these new ports/nets. The ports and nets will have a name appendage added to the name of the PG nets in the top. For example, VSS in the top will be created in the block as VSS__PUSHDOWN.

  By default, a block port becomes obsolete if all cells connected to the that port have been pushed into the block. The command deletes the obsolete port and issues an informational message (limited to 10 by default) when removing the port. To force the tool to retain obsolete ports, specify the block_action retain_obsolete_ports, by using the **set_push_down_object_options** command. If this option is specified, the command will retain these ports as floating ports on the block.

  If multiply instantiated block (MIB) instances exist and cells are specified for push-down into more than one of these instances, the command will push down cells into each instance separately, if the cells are at different locations over the block relative to the block's origin.

  The command has special support for cells which identically overlap different MIB instances. This means that each instance has the same pattern of cell overlap, (such as a chain of repeaters where each chain is overlapping each instance identically and each cell in the chain is the exact same type as a counterpart cell in a chain overlapping another MIB instance). When this is detected by the command, the command pushes the chain into one of the instances, and then essentially reuses that chain for the other instances, and just connects the new port instances for the other MIB instances.

  If a cell being pushed down overlaps more than one block (if the **-blocks** option is used, then if more than one of the specified blocks), then the command reports the partial overlap with all of the blocks that the cell overlaps, and it chooses a block to push down into based upon which block has the most area overlap with the push-down cell. If the area of overlap with multiple blocks is equal for multiple blocks, then the command chooses the block whose boundary contains the push-down cell's origin (assuming the origin lies on top of a pushable block). If the push-down cell's origin is not within the boundary of a pushable block instance, then the command chooses the block whose origin is closest to the cell's origin.

  The valid actions for cells are **set_push_down_object_options -top_action remove** and **set_push_down_object_options -top_action keep** (for physical-only cell pushdown ONLY) and **set_push_down_object_options -block_action copy | create_blockage**.

  If you specify **set_push_down_object_options -block_action copy**, (mutually exclusive with **create_blockage**), the command **push_down_objects** creates a copy of the top-level cell instance in the block. Block ports and nets are created if necessary.

  If you specify **set_push_down_object_options -block_action create_blockage**, (mutually exclusive with **copy**), the **push_down_objects** creates a placement blockage in the block. The netlist is unaffected in both the top and block. The original top-level cell instance is retained in the top block without regard to the -top_action option setting.

If you specify **set_push_down_object_options -object_type cells -allow_multi_rail_cells true**, (only allowed in combination with the **cells** object type), the **push_down_objects** will allow the push-down of MV type cells with multi-rails and no UPF strategy. Without this object-type-specific option, the command will issue an error message and prevent the push-down of cells of this type.

If you specify **set_push_down_object_options -object_type cells -top_action keep**, the command will ONLY push-down physical-only cells, and disregard all other cell type during that push-down session. The physical-only cells created in the blocks will be created with the "copied-down" shadow status, and no nets or ports will be created in the block. The physical-only cells in the top will be untouched.

- **routing_corridor**
  If **routing corridors** are included in the *object_collection*, routing corridors that overlap block boundaries are copied into the block. All nets which are associated with the top-block routing corridor being pushed down are searched for in the block and if a block-level net is found to be connected to the top-level net, then the block-level counterpart gets associated with the new routing corridor in the block. If no block-level nets can be found in the block which map to those in the parent block (by port connection), then the routing corridor in the block will still be created, but with no net associations. If a routing corridor exists in the block with the same name as the one being copied down, and it possesses the copied-down shadow status, it is removed and replaced with the one being copied down.

- **charging_station**
  If **charging_stations** (also referred to as gas stations, which are non-primary voltage areas in MV designs which have associated single-rail repeater cells) are included in the *object_collection*, the command examines the regions of the top-level charging station, and the cells it owns, and determines if the command can legally propagate the voltage area and its cells to the deepest block (in terms of physical hierarchy depth) that the voltage area overlaps. The voltage area must completely fit into that block, and that block must already have UPF supply nets that are equivalent to those associated with the charging station in the top. If the command concludes that it can successfully accommodate the UPF requirements, and the cells are found to be only single-rail repeater buffers or inverters, then the command will create a new charging station in the target block (the deepest overlapped block) and then push the repeater cells down a physical level at a time, propagating all signal ports and nets as it pushes the repeater cells down to the destination target block. It then associates the new cells with the new voltage area and applies all necessary UPF connections and constraints to the new cells and all new ports created on the way down.

  The new charging station will receive all user attributes that existed on the top-level charging station (Voltage Area) and the new repeater cells will also receive copies of user attributes from the top-level repeater cells.

  The charging station is removed from the top if all of its regions were pushed down into the destination block. If only some of the charging station's regions were pushed down, then only those regions are removed from the top block.

  All new charging stations and associated repeater cells are given the appropriate shadow status (pushed-down or copied-down).

  The only valid top action for charging_station objects is *remove*, and the only valid block action for charging_station objects is *copy*. These are also the default top and block actions for *charging_station* objects.

DESCRIPTION

This command transfers objects from the top level to the block level. It will continue pushing down objects to the deepest physical depth as controlled by the command **set_editability** .

The **set_editability** command can enable or disable the **push_down_objects** command for blocks or levels of physical hierarchy. The **push_down_objects** command has no effect on the blocks that are disabled by **set_editability**. It will refrain from pushing further into a block if that block has been marked disabled.

By default, the name marker suffix "__PUSHDOWN" is appended to new cells, ports, and nets created by this command after push-down of routing and cells. You can override this name marker by using the **plan.pins.new_port_name_tag** and/or

plan.pins.new_cell_name_tag application options as follows:

set_app_options -name plan.pins.new_cell_name_tag -value "MYLABEL"
set_app_options -name plan.pins.new_port_name_tag -value "MYTAG"

By default, the tool appends the name marker at the end port or net name. Use the **plan.pins.new_port_name_tag_style** application option to put the name marker at the front of the port name as a prefix.

set_app_options -name plan.pins.new_port_name_tag_style -value "prefix"

Note that this command invokes some setup commands, such as linking blocks and initializing UPF, before it begins to push down objects. For large designs, this can take from a second to minutes. For this reason, it is advisable to run this command on multiple nets instead of one net at a time.

Note that if buses are to be pushed down into blocks as feedthroughs, it is important that all nets of the bus are pushed down in the same call to the **push_down_objects** command, so that the feedthrough ports can be correctly defined as bus ports on the blocks that the bus feeds through. If a net of a bus is pushed down on bit at a time, bus port creation will not be correctly defined for the feedthroughs resulting from bus net push-down.

EXAMPLES

The following example pushes down the routing objects of power and ground nets.

```
prompt> push_down_objects [get_nets {VDD VSS}]
```

The following example pushes down the routing all paths.

```
prompt> push_down_objects [get_shapes]
```

The following example pushes down the routing all paths and vias.

```
prompt> change_selection [get_shapes]
prompt> change_selection [add_to_collection [get_selection] [get_vias]]
prompt> push_down_objects [get_selection]
```

The following example pushes down all placement blockages.

```
prompt> push_down_objects [get_placement_blockages]
```

The following example pushes down all routing blockages.

```
prompt> push_down_objects [get_routing_blockages]
```

The following example pushes down all routing guides into block A.

```
prompt> push_down_objects [get_routing_guides] -blocks [get_cells A]
```

The following example pushes down the buffer BUF1 into block A

```
prompt> push_down_objects [get_cells BUF1] -blocks [get_cells A]
```

The following example pushes down all standalone site rows.

```
prompt> push_down_objects [get_site_rows -filter "is_derived==false"]
```

The following example pushes down all site arrays.

```
prompt> push_down_objects [get_site_arrays]
```

The following example pushes down all site routing corridors.

```
prompt> push_down_objects [get_routing_corridors]
```

The following example pushes down a charging station (or gas station), and all repeater cells belonging to that voltage area that overlap blocks.

```
prompt> push_down_objects [get_voltage_areas cs_00]
```

SEE ALSO

check_objects_for_push_down(2)
get_app_options(2)
help_app_options(2)
pop_up_objects(2)
remove_push_down_object_options(2)
report_push_down_object_options(2)
reset_app_options(2)
set_app_options(2)
set_editability(2)
set_individual_pin_constraints(2)
set_push_down_object_options(2)
write_shadow_eco(2)
plan.pins.new_cell_name_tag(3)
plan.pins.new_port_name_tag(3)
plan.pins.new_port_name_tag_style(3)
plan.pins.new_split_nets_use_block_names(3)
plan.pins.synthesize_abutted_pins(3)

push_rdl_routes

Moves routed redistribution layer (RDL) wires either in a specified direction or away from the specified nets.

SYNTAX

status **push_rdl_routes**
[-nets *collection_of_nets* | -nets_in_file *nets_file*]
[-objects *collection_of_objects*]
-layer *tech_layer_name*
[-mode net | neighbor]
[-direction up | down | left | right]
[-sweep_range *sweep_range*]
[-bounding_box *{ {llx lly} {urx ury} }*]

Data Types

collection_of_nets collection
nets_file string
tech_layer_name string
sweep_range integer
llx float
lly float
urx float
ury float
collection_of_objects collection

ARGUMENTS

-nets *collection_of_nets*

Specifies the RDL nets to be pushed when using the **-mode net** option. For the **-mode neighbor** option, the **-nets** option specifies the net to push routed wires away from.

-nets_in_file *file_name*

Specifies the name of the file that contains the RDL nets to be pushed when using the **-mode net** option.

The **-nets_in_file** and **-mode neighbor** options are mutually exclusive.

-nets has higher priority than **-nets_in_file**.

-objects *collection_of_objects*

Specifies the RDL nets (see **-nets**) by cells or pins. Only the sub-nets which involve the cells or pins are affected.

If the app option **flip_chip.route.routing_style** is **spanning_tree**, the whole nets of the cells or pins are regarded as involved.

You must specify one of these options: **-nets**, **-nets_in_file**, or **-objects**.

-layer *tech_layer_name*

Specifies the metal layer on which to push the specified RDL wires. You can specify only one metal layer. Specify the metal layer by using its technology file layer name.

This option is required.

-mode net | neighbor

Specify the mode used to move RDL nets. If you do not specify this option the command uses **-mode net** by default.

In the net push mode, the command pushes the RDL nets in the direction specified by the **-direction** option. A wire is pushed until it is totally blocked by a routing obstruction (a metal blockage or wire) or it meets the bounding box of the target routed net. When a wire is blocked, the tool splits the wire and tries to push parts of the wire. For example, assume that a vertical wire A is being pushed to the right, as shown in Figure 1, but blockage B partially blocks the path. When the tool finishes pushing the wire, wire A is split into 3 parts: vertical segment A1, horizontal segment C, and vertical segment A2, as shown in Figure 2.

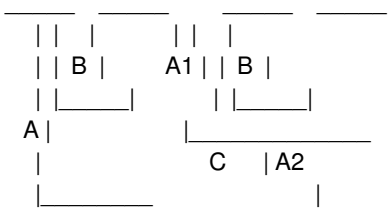


Figure 1 Figure 2

When pushing wires, the tool respects the spacing requirements defined by the **set_app_options {flip_chip.route.layer_bump_spacings list}** command.

In the neighbor push mode, the command pushes the nearest 2xN nets away from a specific net, where N is specified by the **-sweep_range N** option. You can specify only one net at a time.

-direction up | down | left | right

Specifies the direction in which to push the RDL nets.

This option must always be used with the **-mode net** option but never with the **-mode neighbor** option. If you do not specify the direction together with the **-mode net** option, the tool does not push any wires.

-sweep_range *sweep_range*

Specifies the sweep range value used together with the **-mode neighbor** option. The range should be from 1 to 40. Values outside the expected range result in an error. The default is 10.

-bounding_box *{ {llx lly} {urx ury} }*

Specifies a window within which to limit the push operation. For example, assume that you push net A up and you specify a window W as shown in Figure 3. After the push operation, net A is pushed but is kept within window W, as shown in Figure 4.

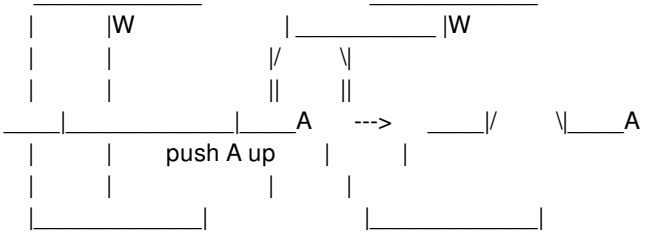


Figure 3 Figure 4

You must also specify the **-mode net** option when you specify the **-bounding_box** option.

The unit for this option is microns.

DESCRIPTION

This command moves routed RDL wires and supports several types of push operations.

When you push an RDL wire by using the **push_rdl_routes** command, the tool moves the wire in the specified direction until the wire is blocked by an obstacle. The command respects the spacing requirements defined by the **create_routing_rule -spacings** command. The RDL wires remain on their original metal layer and the connectivity is not changed.

RDL nets are nets in flip-chip designs or 3DIC designs which are routed on the redistribution layer. In a flip-chip design, RDL nets connect a bump cell to driver I/O cells. In a 3DIC or interposer design, RDL nets can connect the following elements:

- A micro-bump cell and a micro-bump cell
- A micro-bump cell and a TSV (through silicon via) cell
- A micro-bump cell and driver I/O cell

Micro-bump cells are also called flip-chip pads.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command pushes the NET1 and NET2 rdl nets on the METAL7 metal layer in the upward direction.

```
prompt> push_rdl_routes -layer METAL7 -mode net \
  -nets {NET1 NET2} -direction up
```

The following command pushes 40 neighboring nets (20 on each side) away from the NET3 RDL net on the METAL9 layer.

```
prompt> push_rdl_routes -layer METAL9 -mode neighbor \
  -nets NET3 -sweep_range 20
```

SEE ALSO

optimize_rdl_routes(2)
route_rdl_flip_chip(2)
set_app_options(2)
create_routing_rule(2)

query_objects

Searches for and displays objects in the database.

SYNTAX

string **query_objects**
[-verbose]
[-truncate *elem_count*]
[-class *class_name*]
object_spec

Data Types

class_name	string
elem_count	int
object_spec	list

ARGUMENTS

-verbose

Displays the class of each object found.

By default, the command lists only the name of each object. When you use this option, the command lists prefixes the name of each object with its class, as shown in the EXAMPLES section.

-truncate *elem_count*

Truncates the display to *elem_count* elements.

By default, the command displays up to 100 elements. To change the number of elements displayed by the command, use this option. To see all elements, set the *elem_count* value to 0.

-class *class_name*

Specifies the class used to search for objects when an element in the *object_spec* argument is not a collection. Valid classes are application-specific.

If you do not specify this option, the command displays only those elements in the *object_spec* argument that are collections and displays an error message for any other elements, as shown in the EXAMPLES section.

object_spec

Specifies the objects to find and display. Each element in the list is either a collection or a pattern.

- When you specify a collection, the command displays the contents of the collection.

- When you specify a pattern, the command searches the database for objects of the class specified in the **-class** option. If you do not specify the **-class** option, the command issues an error message for the pattern.

DESCRIPTION

The **query_objects** command finds and displays objects in the application's runtime database. The command does not have a meaningful return value; it displays the objects found and returns an empty string.

To control the number of elements displayed, use the **-truncate** option. If the display is truncated, the command prints an ellipsis (...) as the last element. If the default truncation occurs, the command displays a message that shows the total number of elements that would have displayed, as shown in the EXAMPLES section.

Although the output from the **query_objects** command looks similar to the output from any command that creates a collection, the **query_objects** command always returns an empty string.

The **query_objects** command displays the output in either a Legacy format or in a Tcl compatible format. The default output format varies by tool but you can control the format yourself by setting the **query_objects_format** variable. For example, to force Tcl-compatible output use this command:

```
prompt> set_app_var query_objects_format Tcl
Tcl
```

The EXAMPLES section shows output in both the Legacy and Tcl formats.

EXAMPLES

These following examples show the basic usage of the **query_objects** command in both Legacy and Tcl format.

```
prompt> set_app_var query_objects_format Legacy
Legacy
prompt> query_objects [get_cells o*]
{"or1", "or2", "or3"}
prompt> query_objects -class cell U*
{"U1", "U2"}
prompt> query_objects -verbose -class cell \
?   [list U* [get_nets n1]]
{"cell:U1", "cell:U2", "net:n1"}
prompt> set_app_var query_objects_format Tcl
Tcl
prompt> query_objects [get_cells o*]
{or1 or2 or3}
pt_shell> query_objects -class cell U*
{U1 U2}
pt_shell> query_objects -verbose -class cell \
?   [list U* [get_nets n1]]
{{cell U1} {cell U2} {net n1}}
```

When you omit the **-class** option, only those elements of the *object_spec* argument that are collections generate output. The other elements generate error messages.

```
prompt> query_objects [list U* [get_nets n1] n*]
```

```
Error: No such collection 'U*' (SEL-001)
Error: No such collection 'n*' (SEL-001)
{"n1"}
```

When the output is truncated, the command prints an the ellipsis at the end of the display. For the following example, assume the default truncation is 5 (it is actually 100).

```
prompt> query_objects [get_cells o*] -truncate 2
{"or1", "or2", ...}
prompt> query_objects [get_cells *]
{"or1", "or2", "or3", "U1", "U2", ...}
Output truncated (total objects 126)
```

SEE ALSO

collections(2)
get_cells(2)
get_clocks(2)
get_designs(2)
get_generated_clocks(2)
get_lib_cells(2)
get_lib_pins(2)
get_libs(2)
get_nets(2)
get_path_groups(2)
get_pins(2)
get_ports(2)
get_qtm_ports(2)
get_timing_paths(2)

query_pg_extension

Check PG wire shapes meet the requirement that wire shape has desired extension against siterow.

SYNTAX

status **query_pg_extension**
[-net *netname_list*]
[-layer *layer_name*]
[-extension_value *value*]

Data Types

netname_list list
layer_name string
value float

ARGUMENTS

-net *netname_list*

Specifies the name of power or ground nets for wire shape checking. This option combines with **layer** to filter desired shapes. This is a required option.

-layer *layer_name*

Specifies the name of layer for wire shape checking. This option combines with **net** to filter desired shapes. This is a required option.

-extension_value *value*

Specifies a specific value for checking the distance between wire end and siterow

DESCRIPTION

This command checks the distance between wire ends and siterow. If the distance doesn't equal to **extension_value**, it will return the name of the shape.

EXAMPLES

The following example performs checking PG wire extension to siterow.

```
prompt> query_pg_extension \
   -net VDD
   -layer M1
   -extension_value 0.2
```

query_qor_snapshot

Analyzes timing report files from existing QoR snapshots, applies any specified filters, and displays the results in an appropriate format.

SYNTAX

status **query_qor_snapshot**
-name *snapshot_name*
[-directory *directory_name*]
[-type min | max]
[-output_file *file_name*]
[-display]
[-sort_by *column*]
[-group_by *column*]
[-columns *column_list*]
[-and *column_list*]
[-filters *filter_list*]

Data Types

snapshot_name string
directory_name string
column string
column_list list of strings
filter_list string

ARGUMENTS

-name *snapshot_name*

Specifies the name of the Query QoR snapshot result to be analyzed. This is a required option.

-directory *directory_name*

Specifies the name of the directory to search for the Query QoR snapshot specified by the **-name** option.

If you do not specify this option, the tool searches for the QoR snapshot in the directory specified by the application option **time.snapshot_storage_location**.

-type min | max

Loads the appropriate snapshot setup report if you specify **max** and hold report if you specify **min**. This option can only be specified together with the **-name** option.

The default is **max** for loading setup.

/\" .IP "-incremental" /\" Queries the previous queried result instead of querying from scratch. /\" This option and the **-name** option are mutually exclusive. You must /\" specify one of them. /\" /\"

-output_file *file_name*

Specifies the name of the generated output html file.

-sort_by *column_list*

Specifies the column used to sort the results. The default is wns.

To sort a column in descending order, specify "*column_name*" or "+*column_name*". To sort a column in ascending order, specify "-*column_name*". Two exceptions: *wns* and *zero_path* columns have their default sorting order reversed (*+wns* results in an ascending sort). This is so that the default sort order for a column produces the most meaningful sorting order (descending for most numeric columns).

Valid sort columns are: path_group, startpoint, endpoint, scenario, wns, transition_degradation, slew_degradation, zero_path, logic_levels, path_delay, input_delay, output_delay, clock_uncertainty, incremental, transition_time, capacitance, fanout, clock_skew, launch_clock, launch_clock_latency, capture_clock, capture_clock_latency, bufinvcount, io_path, and cross_clock.

-group_by *column_list*

Specifies the column used to group the results. This option groups the results based on the unique values found for this column to create a summary that links to details for that value of the column.

Valid group columns are: path_group, startpoint, endpoint, scenario, launch_clock, capture_clock, bufinvcount, logic_levels, cross_clock and io_path.

-columns *column_list*

Specifies the list of columns to include in the output. The columns are displayed in the specified order. The columns should be separated with spaces. If you specify a column multiple times, its location is determined by the last instance.

Valid columns are: path_group, startpoint, endpoint, scenario, wns, transition_degradation, slew_degradation, derate, zero_path, logic_levels, path_delay, input_delay, output_delay, clock_uncertainty, incremental, transition_time, capacitance, fanout, clock_skew, launch_clock, launch_clock_latency, capture_clock, capture_clock_latency, bufinvcount, cross_clock, io_path, instance_mcnt, libcell_mcnt.

To show all available columns in a predetermined order, specify the **-columns all** option.

Note: You can specify the column names as minimally-distinct strings. That is, instead of using capacitance, you can use "capa" or even "paci", since no other column name contains that sequence of letters.

-filters *filter_list*

Specifies a list of filters to apply to the paths analyzed from the timing report. The specified filters are applied with OR logic; all paths matching any specified filter are selected.

Each filter consists of a filter option and its argument. Most filters accept a range of either floating point or integer numbers as an argument. The argument range consists of two numbers separated by a comma. The first number is the lower bound and the second number is the upper bound. You can also specify just one number. In this case, it is interpreted as the lower bound, unless it is preceded by a comma. If the number is preceded by a comma, it is interpreted as the upper bound.

You can specify one or more of the following filters:

- -wns *lower_bound*,*upper_bound*
  Includes all paths that have a worst negative slack (WNS) within the specified floating point range.
- -transition_degradation *lower_bound*,*upper_bound*

Includes all paths that have their transition degradation ratio (output pin transition time / input pin transition time) within the specified floating point range.

- -slew_degradation *lower_bound*,*upper_bound*
  Includes all paths that have a slew degradation (input pin transition / last stage output pin transition) within the specified floating point range.
- -logic_level *lower_bound*,*upper_bound*
  Includes all paths that have their number of logic levels within the specified integer range.
- -fanout *lower_bound*,*upper_bound*
  Includes all paths that have their worst fanout within the specified integer range.
- -input_delay *lower_bound*,*upper_bound*
  Includes all paths that have an external input delay within the specified floating point range.
- -output_delay *lower_bound*,*upper_bound*
  Includes all paths that have an external output delay within the specified floating point range.
- -clock_uncertainty *lower_bound*,*upper_bound*
  Includes all paths that have clock uncertainty within the specified floating point range.
- -zero_path *lower_bound*,*upper_bound*
  Includes all paths with a zero-path margin within the specified floating point range.
- -clock_skew *lower_bound*,*upper_bound*
  Includes all paths that have clock skew within the specified floating point range.
- -path_delay *lower_bound*,*upper_bound*
  Includes all paths that have a path delay within the specified floating point range.
- -transition_time *lower_bound*,*upper_bound*
  Includes all paths that have a transition time within the specified floating point range.
- -capacitance *lower_bound*,*upper_bound*
  Includes all paths that have their worst capacitance within the specified floating point range.
- -derate *lower_bound*,*upper_bound*
  Includes all paths that have their timing derate within the specified floating point range.
- -incremental *lower_bound*,*upper_bound*
  Includes all paths that have an incremental value within the specified floating point range.
- -bufinvcount *lower_bound*,*upper_bound*
  Includes all paths that have their buffer+inverter count within the specified integer range.
- -io_path *lower_bound*,*upper_bound*
  Include all paths that contain an input or output element in their path. Input/Output elements are: (in), (inout) or (out).
- -cross_clock *lower_bound*,*upper_bound*
  Includes all paths that have different values for the launch and capture clocks. Paths with the same launch and capture clocks will have a value of 0 for this attribute, and paths with different launch and capture clocks will have an attribute of 1.
- -launch_clock *pattern*
  Includes all paths whose launch clock name matches the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern at the beginning or end.
- -capture_clock *pattern*
  Includes all paths whose capture clock name matches the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern at the beginning or end.

- -path_group *pattern*
  Includes all paths whose path group matches the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern at the beginning or end.

- -startpoint *pattern*
  Includes all paths whose startpoint matches the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern at the beginning or end.

- -endpoint *pattern*
  Includes all paths whose endpoint matches the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern at the beginning or end.

- -scenario *pattern*
  Includes all paths whose scenario matches the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern at the beginning or end.

- -instance *pattern*
  Includes all paths that contain an instance matching the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern.

- -file_instance *file_name*
  Includes all paths that contain an instance that matches any of the patterns listed in the specified file (patterns can be separated by space or a newline). Patterns take the same form as those for -instance.

- -libcell *pattern*
  Includes all paths that contain a library cell matching the pattern. You can specify multiple patterns by using the OR operator, "|". Wildcards are supported in the pattern.

- -instance_mcnt *lower_bound*,*upper_bound*
  Includes all paths for which the total count of matches of any of -instance or -file_instance patterns is within the specified range.

- -libcell_mcnt *lower_bound*,*upper_bound*
  Includes all paths for which the total count of matches of any -libcell pattern is within the specified range.

  If you do not specify this option, the following filters are applied:

  -filters "-wns ,0 -zero_path ,0 -fanout 40 -transition_degradation 2.0
  -slew_degradation 2.0 -logic_levels 50"

-and *column_list*

Specifies a list of filters to apply with AND logic. These filters are applied to the paths that pass the filters specified in the **-filters** option. Paths that match one or more OR filters and all AND filters are included in the result.

For example, if you specify **-filters "-wns ,-3.0 -cross_clock 1"**, the command reports the paths with timing violations where WNS is worse than -3.0 OR -cross_clock is true. If you specify **-filters "-wns ,-3.0 -cross_clock 1" -and wns**, the command reports the paths with timing violations where WNS is worse than -3.0 and -cross_clock is true.

DESCRIPTION

The **query_qor_snapshot** command analyzes timing report files from existing QoR snapshots, applies any specified filters, and displays the results in an appropriate format. You can specify which columns to output with the **-columns** option, and which filters to apply as AND filters with the **-and** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

Before running the **query_qor_snapshot** command, you must run the **create_qor_snapshot** command to generate the snapshot data, as shown in the following example:

```
prompt> create_qor_snapshot -name place_opt
```

In the following example, the **query_qor_snapshot** command analyzes a QoR snapshot named preroute, which is saved under the snapshot directory and reports the values with WNS between -2.0 and -1.0.

```
prompt> query_qor_snapshot -name preroute -filters "-wns -2.0,-1.0"
```

In the following example, the **query_qor_snapshot** command analyzes QoR snapshot named placeopt, which is saved under the snapshot directory and reports the values with WNS less than -1.0 or fanout over 40.

```
prompt> query_qor_snapshot -name placeopt -filters "-wns ,-1.0 -fanout 40"
```

The following example shows all paths that have WNS less than zero.

```
prompt> query_qor_snapshot -name timing_report -filters "-wns ,0"
```

The following example shows all paths that have cells with output or input transition time over 2 ns.

```
prompt> query_qor_snapshot -name timing_report \
  -filters "-transition_degradation 2"
```

The following example shows all paths that have path stages with slew degradation over 3 ns.

```
prompt> query_qor_snapshot -name timing_report \
  -filters "-slew_degradation 3"
```

The following example shows all paths that have over 50 logic levels:

```
prompt> query_qor_snapshot -name timing_report \
  -filters "-logic_level 50"
```

The following example shows all paths that have cells with a fanout over 40.

```
prompt> query_qor_snapshot -name timing_report -filters "-fanout 40"
```

The following example looks at all timing paths that match the worst negative slack (WNS) filter or zero-path filter. If the WNS violation is more than -10 ns or the path violates zero-path delay.

```
prompt> query_qor_snapshot -name timing_report \
  -filter "-wns ,-10.0 -zero_path ,0"
```

SEE ALSO

create_qor_snapshot(2)
remove_qor_snapshot(2)
report_qor_snapshot(2)

time.snapshot_storage_location(3)

quit!

quits the application without posting an application exit dialog

SYNTAX

quit!

ARGUMENTS

None.

DESCRIPTION

This command is an alternative to the **quit** command. When called with the GUI up, the **quit** command posts an application exit dialog. To avoid having to "deal with" the exit dialog, the user can instead call the **quit!** command which is guaranteed to exit the application without posting a GUI exit dialog.

EXAMPLES

icc_shell> **quit!**

SEE ALSO

quit(2)

quit

Exits the shell.

SYNTAX

string **quit**

ARGUMENTS

The **quit** command has no arguments.

DESCRIPTION

The **quit** command exits from the application. It is basically a synonym for the **exit** command with no arguments. If there are pending jobs from **redirect -bg, then quit will wait for the jobs to complete before exiting.**

EXAMPLES

The following example exits the current session:

```
prompt> quit
```

SEE ALSO

exit(2)

read_aif

Reads in one AIF (Advanced Input Format) file.

SYNTAX

status **read_aif**
[-use_port_name]
[-ignore_assign_nets]
[-hierarchy]
[-pad_to_ref_list { {*pad_type ref_name orientation*} ... }]
aif_file_name

Data Types

pad_type string
ref_name string
orientation string
aif_file_name string

ARGUMENTS

-use_port_name

Specifies the connection name read from AIF file to be treated as port name. If not specified then the connection name read from AIF file is treated as net name.

-ignore_assign_nets

Specifies the connection names present in AIF file to be ignored. No net connection is made for the bump instances in the AIF file. If not specified then connections are made for the bump instances in AIF file.

-hierarchy

Specifies the IO/BUMPs are read in across all physically hierarchy levels instead of only in top level.

-pad_to_ref_list { {*pad_type ref_name orientation*} ... }

Allows the reference pad type present in AIF file of bump instances to be remapped to a different reference library bump cell. *pad_type* is the reference pad type present in AIF file and *ref_name* is the different reference library bump cell to which remapping needs to be done. The orientation of bump instance can also be changed than what is present in AIF file by using *orientation*. Valid values of *orientation* are N, W, S, E, FN, FW, FS and FE. *orientation* defaults to N if not specified.

aif_file_name

Specifies name of one AIF file to be read.

DESCRIPTION

Reads bump instances from the AIF file and creates them in the current design, if not already present. Net connections of the bump instances are made according to the connections given in AIF file and the bump instances are placed at coordinates as given in AIF file.

The coordinates given in AIF file are relative and are adjusted with die width and height (also present in AIF file) and reference library bump cell coordinates to calculate actual placement coordinate.

If the connection column in AIF file for a bump instance is "-", then the connection of the bump instance is removed.

The reference library bump cell and orientation as given in AIF file can be overridden by using -pad_to_ref_list option to read_aif.

If input AIF file was created with -hierarchy option then AIF file is not read in and a warning message is issued.

EXAMPLES

In the following example, the file my.aif is read and bump instances are created or updated with net connections as given in my.aif

```
prompt> read_aif my.aif
1
```

In the following example, the file my.aif is read and bump instances with reference pad type as BUMPCELL are changed to BUMP2 and the orientation is also changed to south.

```
prompt> read_aif my.aif -pad_to_ref_list { {BUMPCELL BUMP2 S} }
1
```

SEE ALSO

write_aif(2)

read_block_connection_file

Reads in a text file containing information on the connectivity between different blocks, and generates a new top-level design based on that.

SYNTAX

string **read_block_connection_file**
-connections *file_name*
[-design_name *design_name*]
[-map_file *file_name*]
[-ignore_blocks *cell_names*]

Data Types

design_name string
file_name string
cell_names list

ARGUMENTS

-connections *file_name*

Specifies the name of the input text file containing the block connection information in a standard format. This file should be in CSV format and must contain the connection data in the form of columns. The supported standard column names are as follows:

- Module Instance
- Module Reference
- Module Port
- Module Port Direction
- Connected Module Instance
- Connected Module Reference
- Connected Module Port
- Connected Module Port Direction

The first row of the file should contain all these column names. Some of these columns are optional and can be omitted, and some default values will be assumed in this case. For example, "Module Port Direction" and "Connected Module Port Direction" columns are not required and a value of "inout" will be assumed for them if they are missing.

If any of the required columns are missing, the tool will error out. Unsupported columns will be ignored without any error. To use

your own column names, you can provide a map file through **-map_file** option. The columns could be in any order.

-design_name ***design_name***

Specifies the name of the top-level design created from the connection info. If design name is not specified, it defaults to "TOP". The design must not already exist in the current library.

-map_file ***file_name***

Specifies an optional CSV file which maps custom column names in the connections file to standard Synopsys column names in the form <custom_name>,<standard_name> per line.

For example, the map file contents could be as follows:

```
Main_Ref,Module Reference
Other_Port,Connected Module Port
```

In this example, "Main_Ref" and "Other_Port" are custom column names which can be used in the connections file. The custom column names should not be the same as the standard column names. Also, they must not contain any wildcard or special characters.

-ignore_blocks ***cell_names***

Specifies a list of instances to exclude while creating the top-level netlist. Any line in the input file containing one of these block names will be ignored. Therefore, these blocks will not be instantiated in the final created design.

DESCRIPTION

The **read_block_connection_file** command reads in a text file containing information on the connectivity between different blocks and creates a top-level design based on the information. The input file can describe any kind of connectivity between multiple modules and from modules to the top-level ports. For example, Module1 <-> Module2, Module2 <-> Module3, Module3 <-> TOP, and so on. In other words, there is no restriction on how many instances are specified at the top level, how many connections are there between all the instances, or how and which instances are connected to the top-level ports. If some particular instance needs to be skipped from the top-level design, it can be specified via the **-ignore_blocks** option.

Each line in the input file represents one connection, whether it is between modules, from a module to the top-level, or even between two top-level ports. The connection from a module to the top-level port (i.e. the "outside world") can be specified by using special keyword "_TOP_" as the Instance as well as the Reference name. If data for a column is missing, the default will be assumed wherever possible. For example, "inout" will be assumed if the "Module Port Direction" column entry is blank. However, some fields might not have default values, these lines will emit warnings and be ignored if the field is blank or incorrect. The order of the rows is not important as there is no dependency of one connection on another.

The top-level design will be created in the current library. If the current library is not set, the command creates a library using the input file name. The top-level design will be named "TOP", unless another name is provided with the **-design_name** option. The name should not be same as any pre-existing design name. The command saves the design into the current library and sets it as the current design. This can be verified with the **list_blocks** and **current_block** commands. If required, you can write out a Verilog netlist by using the **write_verilog** command or do further analysis on this design.

Data consistency checks

The **read_block_connection_file** utility will do multiple checks on the input data in order to generate a correct and meaningful top-level netlist. Some of these checks are as follows:

1. There should not be any missing or extra column entries in each row, unless a default value can be assumed for that column

2. No column should be repeated.

3. The map file, if provided, must have exactly two columns delimited by comma. The first column should contain custom (non-standard) column names and the second should be one of the standard column names.

4. The entries should not contain any special characters like ? or *.

5. The number of columns should be uniform across all the rows.

6. The directions specified for a multiply-occurring port or pin should be consistent across the file.

7. Instances of two different modules should not have the same name.

8. For a connection to a top-level port, both the Module name and Instance name should be the special keyword "_TOP_".

Any line with incorrect data format in any file will be ignored and a corresponding Warning message will be issued. If the data provided in the input text files is not consistent and clean, the tool will fail to generate the top-level netlist and error out with appropriate Error message.

EXAMPLES

The following example reads in a CSV file with standard columns and generates a top-level design named "TOP".

```
prompt> read_block_connection_file -connections ./test1.csv
Information: Created top-level design 'TOP'. (TLNL-002)
1
```

The following example reads in a CSV file with custom column names, specifies a map file for the column names, specifies the output design name and creates a design in the current library.

```
prompt> read_block_connection_file -connections test2.csv -map_file map.csv -design_name MY_TOP
Information: Setting 'My_Ref' as a custom name for column name 'Module Reference'. (TLNL-017)
Information: Created top-level design 'MY_TOP'. (TLNL-002)
1
prompt> current_lib
{test2}
prompt> list_blocks
Lib test2 /u/user1/test2 current
  *>  0 MY_TOP.design Sep-07-01:51
1
```

SEE ALSO

current_block(2)
current_lib(2)
list_blocks(2)
write_verilog(2)

read_cell_expansion

Reads the cell expansion data from a file to support ASCII flow.

SYNTAX

int **read_cell_expansion**
[-input *input_file_name*]
[-scale_factor *unit_scale*]

Data Types

input_file_name string
unit_scale float

ARGUMENTS

-input *input_file_name*

Specifies the name of the input expansion data file. This is a required argument.

-scale_factor *unit_scale*

Specifies the scale factor between tools. This is optional.

DESCRIPTION

The **read_cell_expansion** command reads the cell expansion data from a file.

EXAMPLES

The following example reads a cell expansion file and applies it to the current design.

prompt> **read_cell_expansion -input in.exp**

SEE ALSO

write_cell_expansion(2)

read_def

Reads in one or more DEF (Design Exchange Format) files.

SYNTAX

status **read_def**
[-design *block*]
[-syntax_only]
[-no_incremental]
[-convert_sites *conversion_spec*]
[-add_def_only_objects all | cells | nets | ports | none]
[-skip_pg_net_connections]
[-traverse_physical_hierarchy]
file_names
[-include *include_list*]
[-exclude *exclude_list*]

Data Types

block string
conversion_spec list
file_names list
include_list list
exclude_list list

ARGUMENTS

-design *block*

Specifies the block into which to read the DEF files.

When you specify this option, the tool does not link the block before reading the DEF files. You must ensure that the block is linked before you run the **read_def** command.

By default, the command reads the DEF files into the current block. If the current block is not linked, the tool automatically links it before reading the DEF files.

-syntax_only

Indicates that the DEF files are processed only to check the syntax and semantics of the files.

-no_incremental

Removes all existing physical annotations from the current block before reading the DEF files. Currently, this option does not support hierarchical designs.

-convert_sites *conversion_spec*

Specifies the mapping of DEF site names using the following syntax:

{ {*from_site to_site*} ...}

The command converts all DEF rows of site *from_site* to rows of site *to_site*.

-add_def_only_objects all | **cells** | **nets** | **ports** | **none**

Specifies the types of objects that are created by the **read_def** command if they exist in the DEF file but not in the current block. You can specify one or more of the following values, except for **none**, which must be used alone:

- **none** (the default)
  The command does not create cells, nets, or ports if they are not in the current block.
- **cells**
  If a cell exists in the DEF file, but not in the current block, the command creates the cell and connects it to the netlist as specified in the DEF file. The command always creates power and ground net connections; The command creates signal/clock net connections if and only if the pins and the net within the same hierarchy. Otherwise, it ignores signal/clock net connections. If the component name contains hierarchy, the hierarchy must already exist in the design; the command does not create new hierarchical cells.
- **nets**
  If a non-PG net exists in the DEF file, but not in the current block, the command creates the net and connects it to the port specified in the DEF PINS section. If the net name contains hierarchy, the hierarchy must already exist in the design; the command does not create new hierarchical cells.
- **ports**
  If a non-PG port exists in the DEF file, but not in the current block, the command creates the port and connects it to the net specified in the DEF PINS section.
- **all**
  This value is equivalent to specifying **cells**, **nets**, and **ports** at the same time.

-skip_pg_net_connections

Indicates that the **read_def** command skips the PG net connections in the DEF files.

-traverse_physical_hierarchy

Indicates reading design information from the DEF files into a hierarchy design.

cells Only physical-only cells can be created in top or sub block when the option "*-add_def_only_objects cells*", is specified at the same time. Other cells only can be placed at the specified location.

nets Only PG nets can be created in top or sub blocks unless the option "*-add_def_only_objects nets*" is specified at the same time. Only unconnected pins can be connected to the net specified in the DEF netlist. All the wiring shapes can be created in the block that is associated with the net.

pins Only top-block pins can be read in the hierarchy design.

via definitions, routing rules and bounds
They are read into sub block only when they are used by wiring or cells in sub block, otherwise, they are read

into top block.

rows, blockages and tracks
They only can be read into top block.

file_names

Specifies one or more DEF files to read.

-include *include_list*

Specifies which optional constructs and object types to include when reading the DEF file. When you use this option, only the listed optional constructs and object types are read. These are the optional constructs and object types:

blockages
bounds
cells
fills
nets
ports
routing_rules
rows_tracks
scanchains
specialnets
vias
diearea

-exclude *exclude_list*

Specifies which optional constructs and object types to exclude when reading the DEF file. When you use this option, only the listed optional constructs and object types are excluded from being read. The valid values for this option are the same as for the "*-include*" option.

-include and *-exclude* are mutually exclusive options.

DESCRIPTION

Reads design information from the DEF files into the specified block or the current block, based on the **-design** option.

If you use relative path names to specify the DEF files, the command uses the **search_path** variable to locate the files. If you use absolute path names to specify the DEF files, the command does not use the **search_path** variable. To determine the file that the **read_def** command reads, use the **which** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reads the DEF file named my.def into the current block. The command uses the **search_path** variable to locate the DEF file.

```
prompt> read_def my.def
1
```

SEE ALSO

current_block(2)
link_block(2)
report_design(2)
remove_blocks(2)
search_path(3)

read_design_io

Reads in a delimited text file that contains location, reference cell, and other information for I/O cells. The command creates or updates the specified I/O elements in the design.

SYNTAX

string **read_design_io**
-file_name_list *file_name_list*
[-map_file *file_name*]
[-delimiter_char *char*]
[-comment_char *char*]
[-c4_ref_name *ref_cell_name*]
[-tsv_ref_name *ref_cell_name*]
[-ubump_ref_name *ref_cell_name*]
[-overlap_check *check_type*]
[-create_ports]
[-power *name_list*]
[-ground *name_list*]
[-die_origin *coordinates*]
[-die_orientation *orientation*]
[-cell_origin_type *string*]
[-units *units_per_micron*]
[-verbose]

Data Types

file_name_list list
file_name string
char string of one character
ref_cell_name string
check_type string
name_list list
units_per_micron integer

ARGUMENTS

-file_name_list *file_name_list*

Specifies the names of one or more input I/O design data files. The files must be ASCII files that contain character delimited values. By default, command is uses as the field separator. A different field separator can be defined via the **-delimiter_char** option. Each row should have exactly same number of fields within a file, however, some fields can be empty in some rows.

-map_file *file_name*

Specifies a CSV file that maps custom column names in the input file to standard Synopsys column names. Custom column names are specified in the first field, and standard Synopsys column names are specified in the second field, separated by a comma.

For example, the following map file maps four custom column names to their corresponding Synopsys column names.

```
IO_driver name,C4_inst
Ref_Cell,Reference_name
X_COORD,X_origin
Y_COORD,Y_origin
```

-delimiter_char ***char***

Specifies a single character to be used as the field delimiter in the output files. If not specified, comma is the field delimiter.

-comment_char ***char***

Specifies a single character to be used as a comment character. When the comment character appears as the first character in a field, that field is ignored.

-c4_ref_name ***ref_cell_name***

Specifies the name of default C4 bump library cell reference to use if the Reference_name field is empty.

-tsv_ref_name ***ref_cell_name***

Specifies the name of default through-silicon via library cell reference to use if the Reference_name field is empty.

-ubump_ref_name ***ref_cell_name***

Specifies the name of micro-bump library cell reference to use if the Reference_name field is empty.

-overlap_check ***check_type***

Specifies the type of check to be done for physical overlap with existing objects while creating new objects. Possible values are **none**, **nominal** and **strict**. Value of **none** means that no checking will be done and a new object will be created even if any other object or cell exists at that location. With value of **nominal**, following rules will apply:

- Two cells of same type cannot overlap with each other.

- C4 bump can overlap with an existing uBump, TSV or a standard cell.

- uBump can overlap with an existing C4 bump, TSV or a standard cell.

- TSV can overlap with an existing C4 bump or a uBump.

Value of **strict** means that no overlap with any existing cell of any type is allowed. This is an optional option and if it is not specified, the default value of **nominal** is assumed.

-create_ports

Create the ports specified in the CSV file if they do not exist at the time of reading the file.

-power ***name_list***

When ports or nets are specified in the input CSV file, if they match the names in the given list, their port_type or net_type attribute respecitively will be set to *power*. If the port or net does not exist in the design, it will be created with type *power*. If the port or net already exists, the port_type or net_type attribute will be updated to *power*.

-ground ***name_list***

When ports or nets are specified in the input CSV file, if they match the names in the given list, their port_type or net_type attribute

respecitively will be set to *ground*. If the port or net does not exist in the design, it will be created with type *ground*. If the port or net already exists, the port_type or net_type attribute will be updated to *ground*.

-die_origin *coordinates*

Specifies the coordinates with respect to which the cells (bumps and TSVs) should be placed. In other words, the specified coordinates will be considered as the die origin. This option is optional. If not specified, the lower left corner of the die (interposer) will be considered.

-die_orientation *orientation*

Specifies the orientation of the Interposer die. If specified, the die is assumed to be rotated accordingly, and hence the coordinates of the newly placed bumps and TSVs will be adjusted appropriately. The allowed values are R0, R90, R180, R270, MX, MXR90, MY, and MYR90. This option is optional. If not specified, the die orientation will be assumed to be R0 (no rotation).

-cell_origin_type *string*

Specifies what point of the cell should be considered as its origin in order to do its placement. The allowed values are *lower_left* or *center*. The bounding box of the cell will be used to calculate its center. This option is optional. If not specified, the lower left corner of the cell will be considered.

-units *units_per_micron*

If the length unit used in the CSV file is different from that of the design library, this option can be used to specify the custom unit for length. When specified, the distance values, i.e. X/Y coordinates specified in the CSV will be divided by this number to derive the distance value in microns. The value must be an integer >=1. This is an optional option. If not specified, the distance values will be assumed to be in the same unit as the current library.

-verbose

Outputs detailed information

DESCRIPTION

The **read_design_io** command reads one or more input files in character delimited format and modifies the current design based on the I/O design data in those files. This command requires an existing NDM library and design to be created and opened, and it will not create a new NDM design. The existing design may or may not already contain design objects, floorplan, etc. Based on the field map and provided I/O object data in the input file, the command can create, place, and connect the following design objects:

- C4 bump cells

- Microbump cells

- Through-Silicon Vias (TSVs)

If a specified I/O object does not exist, it will be created. Otherwise, its placement and connection will be updated based on the file data. The input file is in as ASCII file that is character delimited. The default character delimiter is a comma. A different field separator can be specified with the **-delimiter_char** option. Each row should have exactly the same number of fields within a file, however, some fields can be empty in some rows.

The Following standard columns can be specified in the input file:

- Reference_name

- C4_inst

- Ubump_inst

- TSV_inst

- X_origin

- Y_origin

- Orientation

- Port_name

- Net_name

- Comments

Any additional columns are ignored and the tool issues a warning message. The first row of the input file must be a header row that contains the column names. Each row contains information for one I/O design object. If there are multiple rows for the same object, the later rows will overwrite the previous connection. If the column names in the CSV files are not same as the standard column names specified above, a mapping from the custom column names to these standard names can be provided with the **-map_file** option.

If the reference name for the C4, ubump, or TSV objects are not provided, the default values will be used when creating the objects as defined with the **-c4_ref_name**, **-ubump_ref_name**, and **-tsv_ref_name** options. A specified object reference in the input file will override the default or placement information for that object.

If a cell is already in the design, it will be placed to the location specified by the X_origin and Y_origin fields. If the location data is incomplete (for example, X-origin exists but Y-origin is missing), both columns are skipped. If a cell does not exist, it will be created at the specified location or by default at (0,0). The command will skip the cell and issue a warning if the following are true:

- A reference is not available or cannot be found
- The specified xy coordinates are outside the die bounds
- The specified object overlaps with an existing object

If connectivity is different from the current netlist, the tool will skip the connection information. If connectivity is impossible due to a missing port or some other problem, then no connection will be made.

Any missing, invalid or inconsistent data on a row will cause an error and the I/O object will not be created. If the command encounters an error while parsing the input file, but the "sh_continue_on_error" application variable is "true", the command ignores the row and continues parsing the remaining lines. The supplied data does not have to produce a complete design.

The contents of "Comments" column have no affect on the command. However, ensure that the comment string does not contain a delimiter character (for example, a comma). If a comment character has been specified with the **-comment_char** option, data fields can be defined as comments which will be ignored by the command.

If multiple input files are provided, the subsequent files will overwrite the design information of objects that were specified in previous files. So the order in which the files are specified is important and affects the final result. Different input files can have different number of columns and different order.

Field Map File Details

If the input file contains custom names for fields instead of the Synopsys standard names, a mapping file can be provided via the **-map_file** option. Use the mapping file to create a map between the custom field names and the corresponding Synopsys standard name for the field.

Each line in the map file contains exactly two strings separated by a comma. The first string is a user-defined field string and the second string is a Synopsys keyword. The order of lines does not matter. If the second string is not recognized, the line is ignored with a warning message.

The following map file maps the standard C4_inst field name to "IO_driver", and maps the standard Reference_name field name to "Ref_Cell".

IO_driver,C4_inst
Ref_Cell,Reference_name

The custom field names should not be same as the standard field names. Also, they must not contain any wildcard or special characters.

EXAMPLES

The following example reads in a CSV file with standard column names and creates or updates I/O design objects at the top level.

```
prompt> read_design_io -file_name_list ./test1.csv
Information: 6 lines processed from 1 files.
Information: 0 C4 bumps created. 0 C4 bumps placed.
Information: Location of 1 existing C4 bumps updated.
Information: 0 ubumps created. 0 ubumps placed.
Information: Location of 1 existing ubumps updated.
Information: 0 TSVs created. 0 TSVs placed.
Information: Location of 0 existing TSVs updated.
Information: 0 nets created. 0 net connections made.
Information: 1 C4 bump errors found.
Information: 1 Ubump errors found.
Information: 2 TSV errors found.
Information: 0 syntax errors found apart from the above errors.
1
```

The following example reads in a colon-separated file with custom column names, specifies a map file for the column names, and specifies a default C4 reference lib cell.

```
prompt> read_design_io -delimiter_char ":" -map_file custToSnps.txt \
  -c4_ref_name io.nlib:c4bump.design
Warning: Ignoring entry as number of columns in map file should be exactly 2.
 (custToSnps.txt line 3) (TLNL-014)
Warning: Unrecognized column name 'DUMMY' in map file, ignoring it.
 (custToSnps.txt line 4) (TLNL-018)

Information: 6 lines processed from 1 files.
Information: 0 C4 bumps created. 0 C4 bumps placed.
Information: Location of 1 existing C4 bumps updated.
Information: 0 ubumps created. 0 ubumps placed.
Information: Location of 1 existing ubumps updated.
Information: 0 TSVs created. 0 TSVs placed.
Information: Location of 0 existing TSVs updated.
Information: 0 nets created. 0 net connections made.
Information: 1 C4 bump errors found.
Information: 1 Ubump errors found.
Information: 2 TSV errors found.
Information: 0 syntax errors found apart from the above errors.
1
```

SEE ALSO

create_3d_mirror_bumps(2)
create_bond_pad_array(2)
create_bump_array(2)
create_tsv_array(2)
derive_3d_interface(2)
read_aif(2)
read_block_connection_file(2)
write_design_io(2)

read_drc_error_file

Creates an error data from the given DRC error file.

Please note that, with this command, the error cell is created but not saved. Thus, the error cell has unsaved data. To post-process or overwrite the error cell, use the following commands:

save_block (To save the error cell)

close_drc_error_data -force (To close the error cell)

remove_drc_error_data -force (To remove the error cell)

SYNTAX

status **read_drc_error_file**
[-drc_type *type*]
[-error_data *error_data_name*]
-file *drc_file*

Data Types

type string
error_data_name string
drc_file string

ARGUMENTS

-drc_type *type*

Type of the drc_file. Valid values are: calibre. If this option is not specified, the default type is "calibre".

-error_data *error_data_name*

Name to assign to the error data.

For calibre drc report, if this option is not specified, <cell_name>.err is used by default, where <cell_name> is found inside the drc report file.

If this option is not specified and <cell_name> is not found, this command will abort.

-file *drc_file*

The DRC error report to read in. This is a required option.

DESCRIPTION

The **read_drc_error_file** command takes a DRC error report as input and creates error data. The error data can then be loaded with the error browser.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of how to run the **read_drc_error_file** command.

```
prompt> read_drc_error_file -file drc_report
```

SEE ALSO

signoff_check_drc(2)

read_ivm

Reads via variation file(.ivm) for current desgin.

SYNTAX

status **read_ivm**
file_name
[-corners *corners*]

Data Types

file_name string
corners list

ARGUMENTS

file_name

Specifies the via variation file name to read.

-corners *corners*

Specifies the list of corners for which to apply the extraction options. If corners are not specified then the ivm file will be applied to all corners.

DESCRIPTION

This command reads .ivm file for via variation analysis for the current design.

EXAMPLES

The following example reads new.ivm file for specified two corners.

prompt> **read_ivm new.ivm -corners {FuncMode_max.SS.081v.125c_FuncCmax.corner FuncMode_min.FF.099v.125c_FuncC**

SEE ALSO

report_ivm(2)
remove_ivm(2)
write_ivm(2)
write_parasitics(2)

read_lib_package

Restores the libraries and application option settings from a library package file previously created by the **write_lib_package** command.

SYNTAX

status **read_lib_package**
[-destination *directory_name*]
[-overwrite]
packed_lib_file_name

Data Types

directory_name string
packed_lib_file_name string

ARGUMENTS

-destination *directory_name*

Name of the destination directory where the library will be unpacked. The default is the current directory.

-overwrite

Overwrites any identically named library in the destination directory. By default, an identically named library in the destination directory causes an error, which prevents the library from being restored from the package file.

packed_lib_file_name

The name of the library package file.

DESCRIPTION

This command restores the design library and reference libraries from a library package file previously created by the **write_lib_package** command. After unpacking the library, the command opens the library and sets the current design to match the library that was packed. It also sets the application options and Tcl variables to match the state of the library at the time of packing.

After read_lib_package, please ensure that you have saved all the libraries and blocks onto disk by running "save_lib -all" so all the data from the package persists on disk.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example unpacks the library package file mylib.pack. It restores the design library to the current directory and the reference libraries to a directory called reflibs under the directory of the restored design library.

```
prompt> read_lib_package mylib.pack
1
```

The following example unpacks the library package file mylib.pack and restores the design library to the directory one level above the current directory. If a design library of the same name already exists at the destination, the command overwrites the existing library.

```
prompt> read_lib_package -destination .. -overwrite mylib.pack
1
```

SEE ALSO

write_lib_package(2)

read_name_map

Reads a name map file for the current design.

SYNTAX

string **read_name_map**
name_map_file

Data Types

name_map_file string

ARGUMENTS

name_map_file

Specifies the name of the input name map file.

DESCRIPTION

This command reads the name map data from the specified name map file for the current block.

The name mapping file is a Tcl script containing Tcl built-in commands and two Synopsys commands: **set_query_rules**, which defines renaming rules for rule-based query, and **define_name_maps**, which defines the name mapping for specific objects. The file is provided information only; you should not edit or modify it.

The following is an example of a name mapping file:

```
# Query Rules
set_query_rules hierarchical_separator {/ _} bus_notation {[] __}

# Tcl built-in: local variables
set hier_1  A/B/C/D
set hier_2  A_B/B2/this/is/a/long/path

# Explicit name maps
define_name_maps \
    -application golden_upf \
    -design_name design \
    -columns {class pattern options names} \
    [list cell $hier_1/A_reg [list leaf]       [list $hier_2/ger_A]       ] \
```

```
    [list cell A/B/X*_reg    [list nocase leaf] [list A_B/zar_reg A_B/Y_reg3]]

# cleanup
unset hier_1
unset hier_2
```

EXAMPLES

The following example reads a name map file:

```
prompt> read_name_map top.nmf
```

SEE ALSO

write_name_map(2)
load_upf(2)
set_query_rules(2)
define_name_maps(2)

read_net_estimation_rules

Reads an XML document that contains net estimation rules. This command can be used to re-create the net estimation rules previously saved with the **write_net_estimation_rules** command.

SYNTAX

int **read_net_estimation_rules**
-filename *xml_filename*
[-cell *cell*]

Data Types

xml_filename string
cell collection

ARGUMENTS

-filename *xml_filename*

Specifies the name of the XML document to read that contains net estimation rule data. This command attempts to apply any valid rules and parameters found in the document to the current net estimation rules. The command reports any errors detected while reading the XML document.

-cell *cell*

Specifies the physical cell into which to read the net estimation rule.

By default, rules are read back into the cell in which they were created (the cell attribute in the XML file). If the save file does not specify a cell, they are read into the current block. -cell overrides any saved cell in the XML file.

DESCRIPTION

This command reads an XML document containing net estimation rule data. The XML document can be written by the **write_net_estimation_rules** command or created manually.

NET ESTIMATION RULES AND HIERARCHY

Net estimation rules can be created in any physical block at any level of the hierarchy and multiple blocks can have different rules with the same name. However, rules are often stored/retrieved by name, and doing this can cause confusion as to which rule is being used by a given command. Generally, the "current" (top) block's list of rules is searched by commands such as

estimate_timing. This means that if you define rule R1 in "top" and rule R1 in block "B1", if you make block B1 the current block, commands will use B1's R1. When you go back to top, top's R1 is used. To avoid confusion, use globally unique rule names and define them at the top - they will be propagated to child blocks during distributed commands as necessary. Rules in child blocks that have the same name as rules in top may also be overwritten during distributed commands.

EXAMPLES

The following example creates one non-default rule, writes out the rule as an XML document, and saves the file to rules.xml.

```
prompt> set_net_estimation_rule -parameter register_spacing -value 20 myrule
prompt> write_net_estimation_rules -script rules.xml -format xml
1
```

The following example reads an existing net estimation rule document called rules.xml written out in the previous example.

```
prompt> read_net_estimation_rules -filename rules.xml
1
```

SEE ALSO

report_net_estimation_rules(2)
set_net_estimation_rule(2)
write_net_estimation_rules(2)

read_ocvm

Reads AOCV/POCV derating factor tables.

SYNTAX

int **read_ocvm**
[-corners <list_of_corners>]
ocvm_file

Data Types

ocvm_file string

ARGUMENTS

-corners

Indicates the list of corners for which the AOCV/POCV derate factors are to be applied. If this option is not provided, the corner associated with the current scenario is used.

ocvm_file

Specifies the name of the AOCV/POCV file.

DESCRIPTION

The **read_ocvm** command reads AOCV/POCV derate factor tables from a disk file. The tables are annotated onto one or more design objects. Allowed design object types are hierarchical cells, library cells and designs. The AOCV/POCV data is used to reduce pessimism and improve the accuracy of results. Set the app option **time.aocvm_enable_analysis** to enable AOCV analysis, and set the app option **time.pocvm_enable_analysis** to enable POCV analysis.

Note that a net inherits the AOCV/POCV derate set of a hierarchical cell (design) that fully contains the net. For example, for nets that cross the boundary between the top level and a block (with an associated AOCV/POCV table set), the lowest level hierarchical cell that fully contains the net is not the block. Therefore, the net derating comes from an AOCV/POCV table set that contains the block and fully encloses the net.

When applying AOCV/POCV tables on multiple object types that apply to an arc, the following priority, in decreasing order of precedence, is used to determine the table that applies to the arc:

1. Library cell table
2. Hierarchical cell table

3. Design table

Tables are processed in the order that they appear in the AOCV/POCV file. The last table of a specific type {object_type, rf_type, delay_type, derate_type} to be defined for an object in the file overwrites previous tables defined for the same object of the same type.

The **read_ocvm** command can read binary and compressed binary AOCV/POCV files created using the **write_binary_aocvm** command. No additional arguments are required to read binary or compressed binary AOCV/POCV files.

Multicorner-Multimode Support

By default, this command works on the current corner. To specify a different corner, use the **-corners** option.

EXAMPLES

The following example shows an AOCV derate file, test.aocvm, with a single table annotated on the design.

```
prompt> sh cat test.aocvm

version:       1.0

object_type:    design
rf_type:       rise
delay_type:     cell
derate_type:    late
object_spec:
voltage:       1.02
depth:         1 2 3
distance:      100 1000
table:         1.20 1.10 1.08
               1.22 1.15 1.11

prompt> read_ocvm test.aocvm
```

SEE ALSO

remove_ocvm(2)
report_ocvm(2)
report_timing(2)
time.aocvm_enable_analysis(3)
time.pocvm_enable_analysis(3)
time.ocvm_enable_distance_analysis(3)
time.ocvm_precedence_compatibility(3)
time.aocvm_analysis_type(3)
time.aocvm_enable_clock_network_only(3)

read_optimization_history

Read design optimization history from saved file.

SYNTAX

status **read_optimization_history**
[-file *file_name*]

DESCRIPTION

This command restores the history of the optimization steps applied to the design.

SEE ALSO

read_compile_history(2)
write_compile_history(2)
report_optimization_history(2)

read_parasitic_tech

Reads in one or more parasitic model files in TLUPlus format or common nxtgrd file format.

SYNTAX

string **read_parasitic_tech**
-tlup *tlup_files*
[-layermap *layer_map_file*]
[-name *name*]
[-sanity_check none | basic | advanced]
[-fill_emulation true | false]

Data Types

tlup_files string
layer_map_file string
name string

ARGUMENTS

-tlup *tlup_files*

Specifies the names of one or more TLUPlus files or common nxtgrd file to read. This argument is required and you must provide at least one TLUPlus file name.

-layermap *layer_map_file*

Specifies the names of the layer map file between the technology file and the ITF file.

-name *name*

Specifies the name which identifies this model. If not specified, the base file name is used as the parasitic tech model name.

-sanity_check none | **basic** | **advanced**

Disables or enables certain level of sanity check on TLUPlus file and layer mapping file. Default value is basic. Set value "none" to disable all sanity checks. Set value "advanced" to enable strict layer mapping file checks.

-fill_emulation true | **false**

Read TLUPlus file with or without fill emulation. Default value is false. Use this option for the common tech file only. Set value "false" to read TLUPlus file without fill emulation. Set value "true" to read TLUPlus file with fill emulation.

RC parasitic extraction recognizes the parasitic tech by model name. A different name implies a different parasitic tech.

DESCRIPTION

This command reads in parasitic model data files. After the data is read in, it is stored in current library and the TLUPlus files or common tech file are no longer referenced. The data and TLUPlus files name are saved to disk along with other library data when the **save_lib** command is issued. The parasitic data is used in RC parasitic extraction, delay calculation, net estimations, and other applications.

The layer map file maps the technology file layer name to Interconnect Technology Format (ITF) layer name. The "conducting_layers" and "via_layers" keywords are used to identify the layer properties. Lines starting with * or # are comments in the layer mapping file. If there is no layer map file specified, the ITF layer name is treated the same as the technology layer name.

The following is an example of the layer map file.

```
*****Technology_maskName VS ITF_layerName*****
conducting_layers
poly     fpoly
metal1   metal1
metal2   metal2
metal3   metal3
metal4   metal4
metal5   metal5
metal6   metal6
metal7   metal7
via_layers
polyCont  polyCont
via1     via1
via2     via2
via3     via3
via4     via4
via5     via5
via6     via6
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example reads in the file.tlup TLUPlus file.

```
prompt> read_parasitic_tech -tlup file.tlup -layermap file.layermap \
  -name critical_worst_param_1
```

The following example reads in the file.nxtgrd nxtgrd file.

```
prompt> read_parasitic_tech -tlup file.nxtgrd -layermap file.layermap \
  -name critical_worst_param_1
```

SEE ALSO

all_corners(2)
create_corner(2)
create_mode(2)
current_corner(2)
get_corners(2)
get_parasitic_techs(2)
remove_corners(2)
save_lib(2)
set_parasitic_parameters(2)
set_scenario_status(2)

read_parasitics

Reads parasitics from SPEF files or GPD path, GPD attachment in ndm design for current design and sub designs.

SYNTAX

status **read_parasitics**
[-corner_spef {*corners spef_file* [*corner_type*]}\
[-gpd gdp_path]
[-gpd_attach]
[-block *block_name*]
[-validate]

Data Types

corners list
spef_file string
corner_type string
block_name string
gpd_path string

ARGUMENTS

-corner_spef {*corners spef_file corner_type*}

Specifies the SPEF files associated with the constraint corners.

You can specify this option multiple times. If corners share the same SPEF file, you can specify them in the same **-corner_spef** option by listing the corner names. Otherwise, you must specify the files in different **-corner_spef** options. See the first example in the EXAMPLES section.

The valid values for the *corner_type* argument are **early** and **late**. If you do not specify the *corner_type* argument, the SPEF file is for both early and late.

The .spef_scenario file generated by the **write_parasitics** command has the association of corner scenarios with the SPEF files.

-gpd *gpd_path*

Specifies the gpd path for RC annotation. Users need to make sure the input GPD which have the matched corners with corners defined in the working block. Otherwise, read parasitics from GPD would be failed with error message.

-gpd_attach

read RC from gpd attached in the working ndm design. If ndm design does not have gpd attached, read parasitics from GPD attachment would be failed with error message.

-block *block_name*

Specifies the block name associated with the specified SPEF files. Use this option if the SPEF files are for a subblock.

If the SPEF files are for a subblock, the corners specified in the **-corner_spef** options should be corners defined for the top-level block.

By default, the SPEF files are for the top-level block.

-validate

Execute validation.

DESCRIPTION

This command reads parasitics for the current design and sub designs from the SPEF files, or GPD path, GPD attachent. **read_parasitics** is the block level command. You need to specify one **read_parasitics** command for each block for hierarchical design. The **read_parasitics** command would annotate RC for the nets reading from SPEF files,or GPD path, GPD attachment. The nets in the design not annotated would be estimated when executing the **report_timing** command after running **read_parasitics**.

EXAMPLES

The following example reads SPEF files for the current block, which has multiple constraint corners. Two constraint corners of the total eight corners share one SPEF file.

```
prompt> read_parasitics \
  -corner_spef {{FuncMode_max.SS.081v.125c_FuncCmax.corner \
    FuncMode_min.FF.099v.125c_FuncCmax.corner} dr_cc_max_125.spef} \
  -corner_spef {{FuncMode_min.FF.099v.m40c_FuncCmax.corner \
    FuncMode_max.FF.081v.m40c_FuncCmax.corner} dr_cc_max_m40.spef} \
  -corner_spef {{FuncMode_max.SS.081v.125c_FuncCmin.corner \
    FuncMode_min.FF.099v.125c_FuncCmin.corner} dr_cc_min_125.spef}
  -corner_spef {{FuncMode_min.FF.099v.m40c_FuncCmin.corner \
    FuncMode_min.FF.081v.m40c_FuncCmin.corner} dr_cc_min_m40.spef}
```

The following example reads SPEF files for a hierarchical design that has one subblock. The first **read_parasitics** command is for the top-level block. The second **read_parasitics** command is for the subblock.

```
prompt> read_parasitics -corner_spef {s1.corner input/top.spef}
prompt> read_parasitics -block {rm2000_l2data_spr4096x288_bistw} \
  -corner_spef {s1.corner input/block.spef}
```

The following example reads GPD path for a current block

```
prompt> read_parasitics -gpd input/top.gpd fP
```

The following example reads GPD attached in current ndm block for a current block

```
prompt> read_parasitics -gpd_attach fP
```

SEE ALSO

all_corners(2)
report_extraction_options(2)
report_parasitic_parameters(2)
report_timing(2)
set_extraction_options(2)
set_parasitic_parameters(2)
write_parasitics(2)

read_physical_rules

Imports library/cell level abstract physical rules and attributes through PRF file to target library and its cells.

SYNTAX

int **read_physical_rules**
file_name
[-include *include_list*]
[-library *lib*]

Data Types

file_name string
lib string
include_list list of string

ARGUMENTS

***file_name*:**

Specifies the file name to be read in. This is required option.

-include *include_list*

Specifies which rules to be imported from the input file.

- **library**: Imports library level rules.
- **cell**: Imports cell/pin level rules and attributes.
- **instructions**: Imports stream in instructions as corresponding app-options. This option is avaiable only in library manager.
- **site**: Imports site and row_pattern and track_pattern under these two groups.
- **layer**: Imports layer definition under library level.
- **pin_attr**: Imports pin level attributes such as "direction","port_type", "pg_type" and "is_secondary_pg". This option can only work together with "cell".
- **routing_blockage**: Imports routing_blockage definition. This option can only work together with "cell".
- **all**: Imports all rules and attributes.

  The default value in library manager is *{library cell}*, in ICC2 is *{library}*.

-library *lib*

Specifies the target library to import the rules and attributes. This option is only avaiable in ICC2. If not specified, the target library will be found by the library name in PRF file, or current library if not found.

In library manager, it's always the current worksapce library.

DESCRIPTION

This command reads in abstract physical rules and attributes from input file to target library and all its cells. In library manager, the file must be read in before **check_workspace**, as cell data will be saved only at that time. In ICC2, the file can be read to runtime ref-libs by specifying -library.

EXAMPLES

The following example reads in all rules from in.prf to current library and all its cells.

```
prompt> read_physical_rules in.prf -include all
2
```

The following example reads in instructions an cell attributes/rules from in.prf to current library.

```
prompt> read_physical_rules in.prf -include {cell instructions}
2
```

The following example reads in cell level rules from in.prf to libA.

```
prompt> read_physical_rules in.prf -include cell -library libA
2
```

SEE ALSO

write_physical_rules(2)
remove_physical_rules(2)

read_pin_constraints

Reads pin constraints from the specified file and applies them to the current design.

SYNTAX

status **read_pin_constraints**
[-file_name *file_name*]

Data Types

file_name string

ARGUMENTS

-file_name *name*

Specifies the file name that contains the pin constraints. The constraints are used by the router and pin placer.

DESCRIPTION

This command reads a file which contains pin and feedthrough constraints. The pin constraint file contains multiple sections that contain different types of constraints for the router and pin placer. Four sections are supported: topological map, physical pin constraints, pin spacing control, block pin constraints. Pin-based physical pin constraints supports multi-level physical hierarchy (MPH).

This command stores the pin constraints in the database. The constraints are treated the same as constraints that are set with other Tcl commands. You can use the **report_individual_pin_constraints** and **remove_individual_pin_constraints** commands to report and remove pin constraints.

The **set_editability** command can enable or disable this command for blocks or levels of physical hierarchy. This command has no effect on the blocks that are disabled with the **set_editability** command.

Each section begins with a section header. Supported section headers are "start topological map", "start feedthrough control", "start physical pin constraints", "start pin spacing control", "start block pin constraints;". Each section ends with a section footer and a semicolon (;). The section footer corresponds to the section header and begins with the "end" keyword, for example "end topological map;". The constraint file specifies how nets should be routed through blocks in routing stage and how pin should be placed during pin placement.

Empty lines or lines with only white spaces are ignored. Leading or trailing white spaces are ignored. Lines that begin with comment character (#) are comments and are ignored.

Curly brackets ({}) are used to separate different constraints and objects. A semicolon (;) separates different records. If there is more

than one object following a constraint keyword, you must use curly brackets. In the following example, the first layer constraint is incorrect because the layer names are not surrounded by curly brackets. The second constraint shows the correct syntax.

```
{{cell A} {layers M2 M4}}  # incorrect
{{cell A} {layers {M2 M4}} # correct
```

Topological Map

The topological map section contains routing topologies for different nets in the design. Add constraints to this section to specify how nets should be routed through blocks during pin placement.

Each record contains one or more net or bundle names, followed by pairs of linked objects. Use curly brackets to specify more than one net name. The objects can be block cell, top-level port or any cell instance pin. The pin can be a macro pin or standard cell pin, but cannot be the pin on physical block. Note that ports and pins cannot be specified for bundles. In following example, a net connects to cell_A, Cell_C, and Cell_E, you can force the routing to go through Cell_B to connect Cell_A and Cell_C as follows:

```
start topological map;
{Nets Net_A}
  {{{Cell Cell_A} {Cell Cell_B}}
   {{Cell Cell_B} {Cell Cell_C}}
   {{Cell Cell_A} {Cell Cell_E}}};
end topological map;
```

The following example forces routing to pass through Cell_B to connect top-level port A with a macro or standard cell pin C/a if Net_A does not connect to Cell_B:

```
start topological map;
{Nets Net_A}
  {{{Port A} {Cell Cell_B}}
   {{Cell Cell_B} {Pins C/a}}};
end topological map;
```

When you have a set of nets with the same routing patterns, you can describe them as a group of nets within curly brackets separated by spaces. In addition, you can use "*" as a wildcard to specify a set of nets, such as a bus. Note that when a set of nets are specified, you cannot use top-level ports or pins to describe the connectivity.

```
start topological map;
{Nets {Net_A Net_B DataIn*}}
  {{{Cell Cell_A} {Cell Cell_B}}
   {{Cell Cell_B} {Cell Cell_C}}};
end topological map;
```

For block objects, you can specify side, offset, and layer constraints for the block pins. The side specifies the side number for a block instance where the pins are inserted. The side number is a positive integer that starts from 1. Given any block instance with a rectangular or rectilinear shape, the leftmost edge is side number

1. If there are multiple leftmost edges, then edge 1 is the lowest leftmost edge. The sides are numbered in consecutive order proceeded clockwise around the shape. The shape here referring to block instance's shape, not the reference block's shape.

The offset specifies the distance in microns from the starting point of a specific edge to the routing cross-point (or the block pin's center location) on that edge in clockwise direction. The starting point is also following the clockwise fashion. For instance, for a bottom horizontal edge, the starting point is the right vertex of the edge as it is counted in clockwise and bottom edge's starting point is right point vertex. Since the offset is specific on an edge, you must specify the side information with your offset. The edge here is referring to the edge on block instance, not the reference block. The topological constraints not support negative offset value.

The layer constraint specifies the metal layers to use for the pins.

The following example shows one net record in the topological map section that contains side, offset, and layers of the pins. The routing connects Cell_A through side 1 at offset 20.18 on layer M2 to Cell_B through side 3 at offset 40.28 on layer M4. The routing also connects Cell_B through side 1 at offset 30.28 on layer M4 to Cell_C through side 3 at offset 15.88 on layer M2.

```
start topological map;
{Nets Net_A}
  {{{Cell Cell_A} {sides 1}
    {offset 20.18} {layers M2}}
   {{Cell Cell_B} {sides 3}
    {offset 40.28} {layers M4}}}

  {{{Cell Cell_B} {sides 1}
    {offset 30.28} {layers M4}}
   {{Cell Cell_C} {sides 3}
    {offset 15.88} {layers M2}}};
end topological map;
```

In addition, for block objects, you can specify routing cross-point's location range (or block pin's location range) on a side. The location range is specified with a pair of numbers within {}. In following example, the routing will route from Cell_A through side 2 with location range between 10 micron offset and 20 micron offset when connecting to Cell_B, and routing will route through Cell_B's side 4 with location range between 10 micron offset and 20 micron offset.

```
start topological map;
{Nets Net_A}
  {{Cell Cell_A} {sides 2}
    {offset {10 20}}}
  {{Cell Cell_B} {sides 4}
    {offset {10 20}}};
end topological map;
```

For bundle-based topological constraints, the wildcard character (*) is not supported. For instance, you cannot specify a wildcard character for bundles, ports or pins as the bundle is a group of nets. In this context, the wildcard character for bundles, ports or pins can be confusing.

For a very complicated routing topology which involves multiple enters/exits on same block instance, the tool might not be able to generate routing topology that user intends to. In this case, it is recommended to start from the driver to define the routing topology. In the following example, a routing topology started from driver block A and goes to block B with side 1/offset 20, and then goes from block B to load block C. On the separated routing path on same net, the routings branch out from driver block A and goes to block B again and then goes to load block D. With different offset (40) on the same side(1) of block B, the topological constraints can be specified in following ways:

```
start topological map;
{Nets Net_E}
  {{Cell A} {{Cell B} {sides 1}{offset 20}}}
  {{{Cell B} {sides 3}{offset 40}} {Cell C}}
  {{Cell A} {{Cell B} {sides 1}{offset 40}}}
  {{{Cell B} {sides 3}{offset 20}} {Cell D}}
end topological map;
```

When there are multiple enters/exits on block B, in order to avoid ambiguity, it is recommended to define the topology starting from driver. Also the physical constraints (side/offset or offset range} can be used to help resolve the ambiguity.

Feedthrough Control

The feedthrough control section enables or disables feedthroughs for specific nets and blocks. Each record begins with the Nets keyword and contains one or more net names within curly brackets. The net names are followed by "Feedthrough" or "NoFeedthrough" keywords, followed by block cells within curly brackets.

The following rules are used when applying the feedthrough control:

- Later statements override earlier statements for the same nets.
- The Feedthrough and NoFeedthrough keywords specify whether feedthroughs are allowed or not by default, for either specified block cells or for the entire design. If block cells are specified, all other blocks cells assume the opposite feedthrough constraint value. Therefore, only one Feedthrough or NoFeedthrough constraint line is required per specification, there is no need to specify both the Feedthrough and NoFeedthrough keywords in a single feedthrough specification.
- Topological map constraints can overrule feedthrough control statements. For example, the following topological map constraint forces a feedthrough on block B, regardless of whether feedthroughs are allowed on B.

  {{{cell A} {cell B}} {{cell B} {cell C}}}

The following example enables feedthroughs for net Xecutng_Instrn[0] on blocks I_ALU, I_REG_FILE and I_STACK_TOP.

```
start feedthrough control;
{Nets Xecutng_Instrn[0]}
  {Feedthrough {I_ALU I_REG_FILE I_STACK_TOP}};
end feedthrough control;
```

The following example prevents feedthroughs for net Xecutng_Instrn[0] on blocks I_ALU, I_REG_FILE and I_STACK_TOP.

```
start feedthrough control;
{Nets Xecutng_Instrn[0]}
  {NoFeedthrough I_ALU I_REG_FILE I_STACK_TOP};
end feedthrough control;
```

A single Feedthrough or NoFeedthrough statement completely specifies the allowed feedthrough modules for the net. Feedthrough statements are not cumulative. If multiple feedthrough statements exist for the same net, the latest feedthrough statement take over the previous statement. This rule simplifies the organization of the feedthrough file and prevents complex interactions between feedthrough statements.

Feedthrough and NoFeedthrough statements override the default feedthrough permission set by the **set_block_pin_constraints -allow_feedthroughs true|false** command for the specified nets. However, feedthrough allow/disallow set by **set_individual_pin_constraints -allow_feedthroughs true|false** command for individual net will overrule the Feedthrough/NoFeedthrough statements.

The statement "{ Feedthrough A B }" does not force the creation of feedthroughs on A and B. Instead, the statement allows feedthroughs on blocks A and B, while restricting feedthroughs on all other blocks. To force a specific feedthrough path you must use the pairwise statements described later.

If there are feedthrough constraint conflicts between topological map constraint and previous feedthrough constraints, the feedthrough constraints from constraint file take priority. If there are conflicts between feedthrough constraints and topological map constraints, topological map constraints take priority.

Physical Pin Constraints

The physical pin constraints section specifies physical pin constraints for individual nets, pins or ports. Supported keywords are: net, pins, reference, sides, layers, offset, order, start, end, off_edge, location, width and length.

The syntax for most supported constraint keywords (sides, layers, width, length, etc) is the same as in the command **set_individual_pin_constraints**. See the related man pages for more details. For offset, **read_pin_constraints** can take a number as well as a range, which includes a start point and an end point.

Each record in the physical pin constraints section is a set of key words with values enclosed by curly brackets and ended by a semicolon. The first element in the record can be a top-level net or a port for a reference block. If the record is a port object, the block is expected to follow with reference block's name associated with the port.

The physical pin constraints section also supports order for a collection of pins or ports, which is not very convenient by using command **set_individual_pin_constraints**. To set the order for a collection of pins or ports, it must specify the desired edge. And it is also a good practice to specify the range by using **start** and **end**. Otherwise, it means the whole edge. In the most simple case as shown in the following example, the distance between the adjacent order constrained pins or ports is roughly equal to the range divided by the number of order constrained pins or ports. Therefore, if there is a big blockage taking away a large chunk of edge, the range should reflect the existence of such a blockage. Generally, the order constraints are intended for a collection of pins or ports on a region with continuous available empty wire tracks for pin placement.

If a specific order plus an exact spacing are desired for a set of pins, the recommended approach is to define a bundle and constrain the pin order and spacing with **set_bundle_pin_constraints**. Please refer to manpages of **create_bundle** and **set_bundle_pin_constraints**.

The following example applies constraints on a design with two block instances **Inst1** and **Inst2**, whose reference block names are **BlockA** and **BlockB**, respectively.

The first two records constrain pins on net **E** connecting block instances **Inst1** and **Inst2**. The next three records constrain the pins **C**, **D** and **F** on reference block **BlockA**. The last record set the order of pins **in[1]**, **in[2]** and **in[3]** on reference block **BlockB**.

```
start physical pin constraints;
{Net E} {cell Inst1} {layers METAL2} {sides 2};
{Net E} {cell Inst2} {layers METAL4} {sides 4};
{Pins C} {reference BlockA} {layers METAL2} {sides 1} {offset {40 50}};
{Pins D} {reference BlockA} {layers METAL2} {width 0.8} {length 0.8};
{Pins F} {reference BlockA} {off_edge} {location {10 10}};
{reference BlockB} {sides 2} {start 10} {end 80} {order {in[1] in[2] in[3]}};
end physical pin constraints;
```

The side constraint specifies the side number on which the pin should be inserted for the specified reference block. The side number is a positive integer that starts from one. Given any rectangular or rectilinear shape, the leftmost edge is side number one. If there are multiple leftmost edges, then edge one is the lowest leftmost edge. The sides are numbered in consecutive order proceeded clockwise around the shape. The shape is referring to the reference block's boundary shape.

The offset constraint specifies the distance in microns from the starting point(positive value) or ending point (negative value) of a specific edge to the routing cross-point on thate dge in the clockwise direction. The starting point is also followingt he clockwise fashion. For instance, for a bottom horizontal edge, the starting point is the right vertex of the edge as it is counted in clockwise and bottom edge's starting point is right point vertex, vice versa, for ending point is the left vertex of the edge. Since the offset is specific on an edge, you must specify the side information with your offset. The edge here is referring to the reference block's boundary edge.

The location constraints value are referred to the reference cell in the pin constraint entry and therefore, the coordinates will translated to top-level view.

A more complicated order constraint is shown in the following example. The pin order **A** and **B** is specified as to be 2 wire tracks in between with **A** first and **B** second. **B** also must use layer M4. Pins **A**, **B**, **C** and **D** must be placed within the region between 20 to 40 microns on side 3 of block **RISC_CORE**. Because pin **C** and **D** are not specified within the {order...} constraint, they can be anywhere within the 20 to 40 micron region on side 3.

```
start physical pin constraints;
{reference RISC_CORE}
  {sides 3} {start 20}
  {end 40} {{order {A {spacing 2} {layers METAL4} B}} {pins {C D}}};
end physical pin constraints;
```

If there are multiple records constraining the same object with conflict settings, the last record overwrites the previous ones.

Finally, pin constraints can be applied by either one of the commands **set_individual_pin_constraints**, **set_bundle_pin_constraints**, or **read_pin_constraints**. But try to use the same method to apply the constraints if possible so as to avoid unwanted conflict between different methods. If unclear whether or not the database was applied constraints before, or the type of the constraints applied, use **remove_individual_pin_constraints** or **remove_bundle_pin_constraints** to clear them.

Pin Spacing Control

The pin spacing control section specifies pin spacing constraints for individual edges on each block cell.

Each record specifies the pin spacing constraints for one or more edges on one or more layers for a particular block cell. The constraint contains the reference, side, and layer keywords. Keywords are case insensitive.

In addition, for those edges specified in the pin spacing control section, those layers that are NOT listed are NOT allowed for pin placement. Furthermore, if those listed layers are NOT allowed in the corresponding reference blocks, they are NOT allowed for pin placement either.

```
start pin spacing control;
{reference ref_cell_name} {sides {side_a side_b}}
{layer_a spacing_a} {layer_b spacing_b};
end pin spacing control;
```

The reference keyword specifies the block reference that this constraint applies to. For a block cell, this constraint refers to the reference cell name, not the cell instance name, if it is to constrain the top level.

The sides keyword specifies one or more sides of the block. If there is more than one side, specify the side number within curly brackets separated by white space. You can use a wildcard character (*) to specify that all sides are allowed.

After the side constraint, layer and spacing constraints are specified as pairs. You can specify multiple layer-spacing pairs. The layer value is the layer name. The spacing value is the minimum number of wire tracks between adjacent pins for the specified layer.

In the following example, the CPU block cell is constrained to minimum pin spacing of 2 wire tracks on layer M2, a minimum pin spacing of 3 wire tracks on layer M4, and minimum pin spacing of 3 wire tracks on layer M6 for sides 2 and 4 of the block:

```
start pin spacing control;
{reference CPU} {sides 2 4} {METAL2 2} {METAL4 3} {METAL6 3};
end pin spacing control;
```

Keywords are case insensitive. The reference name and the layer name are case-sensitive.

If a line contains a syntax error, the line is ignored. The command issues warnings or errors that report any lines that are ignored.

If the pin spacing control section contains conflicting constraints for the same edge, same layer and the same block, the last constraint takes priority.

Block Pin Constraints

The block pin constraints section specifies pin constraints that apply to all the pins of the specified block cell.

You can specify block-related pin constraints for feedthroughs, layers, spacing, corner keepouts, hard pin constraints, sides and exclude sides. The same constraint can be set with the **set_block_pin_constraints** Tcl command. Supported keywords are: reference, sides, exclude_sides, layers, feedthrough, spacing, spacing_distance, corner_keepout_distance, corner_keepout_num_tracks and hard_constraints.

The following example sets pin constraints:

```
start block pin constraints;
{reference DATA_PATH}
 {layers {METAL3 METAL4 METAL5}}
 {feedthrough true}
 {corner_keepout_num_tracks 1}
 {spacing 5}
 {exclude_sides {2 3}};

{reference ALU}
```

```
 {layers {METAL3 METAL4 METAL5}}
 {feedthrough true}
 {hard_constraints {spacing location layer}}
 {sides {2 3}};
end block pin constraints;
```

You can use the **write_pin_constraints** command to write out block cell pin constraints to a file, and use the **read_pin_constraints** command to read the file.

Bundle Pin Constraints

The bundle pin constraints section specifies bundle pin-related constraints. This section is now included in physical pin constraints section.

You can specify whether bundle pin constraints are honored or not during pin placement. The same constraints can be set with the **set_bundle_pin_constraints -keep_pins_together true|false** command. The following example sets bundle pin constraints:

```
start physical pin constraints;
{bundle bundle_0} {keep_bundle_pins_together true};
end physical pin constraints;
```

You can use the **write_pin_constraints** command to write out the bundle pin constraints to a file, and use the **read_pin_constraints** command to read the file.

Conflicts

When there is a conflict between a topological map constraint and a physical pin constraint, topological map constraints takes priority over physical pin constraints for global routing and pin placement.

The constraint file can have multiple topological map sections or multiple physical pin constraints sections. If same object is listed multiple times with the same constraint, the last record takes priority. If the same object is listed multiple times with different constraints, the constraints are combined.

EXAMPLES

The following example reads the pin_cstr pin constraint file.

```
prompt> read_pin_constraints -file_name pin_cstr
```

A complete pin constraints file example is shown below.

```
start topological map;
{nets net_B}
{{{cell Cell_A} {sides 1}
  {offset 20.18} {layers M2}}
 {{cell  Cell_B} {sides 3}
  {offset 40.28} {layers M4}}
 {{cell  Cell_C} {sides 1}
  {offset 30.28} {layers M4}}
 {{cell  Cell_D} {sides 3}
  {offset 15.88} {layers M2}}};
end topological map;

start feedthrough control;
{nets Xecutng_Instrn[0]}
```

```
  {NoFeedthrough {I_ALU I_REG_FILE I_STACK_TOP}};
end feedthrough control;

start physical pin constraints;
  {pins C} {reference B} {layers METAL2 METAL4} {sides 2};
  {pins D} {reference B} {sides 2};
  {net C/A} {layer METAL4};
  {bundle bundle_0} {keep_bundle_pins_together true}
end physical pin constraints;

start pin spacing control;
{reference C} {sides 2 4} {METAL2 2} {METAL4 3} {METAL6 3};
end pin spacing control;

start block pin constraints;
{reference DATA_PATH}
  {layers {METAL3 METAL4 METAL5}}
  {feedthrough true}
  {corner_keepout_num_tracks 1}
  {spacing 5}
  {exclude_sides {2 3}};

{reference ALU}
  {layers {METAL3 METAL4 METAL5}}
  {feedthrough true}
  {hard_constraints {spacing location layer}}
  {sides {2 3}};
end block pin constraints;
```

SEE ALSO

remove_individual_pin_constraints(2)
report_individual_pin_constraints(2)
set_individual_pin_constraints(2)
set_editability(2)

read_rde

Reads RDE model from a encrypted file for current design.

SYNTAX

status **read_rde**
file_name
[-help]

Data Types

file_name string

ARGUMENTS

file_name

Specifies the name of the input file from which a RDE model for the current design are read.

-help

Specifies the usage of read_rde

DESCRIPTION

This command reads the RDE model for the current design from a file. The file is an encrypted format generated by the "write_rde" command. For very large designs, the runtime of the RDE capture step can be reduced by dumping the RDE model from a prior run with "write_rde" and reading it back into a successive run with "read_rde". Once the RDE model information is read in, before either the place_opt or clock_opt flow stages, the subsequent RDE capture step(s) will be skipped subject to the following use model. A RDE model dumped after place_opt (final_opto) may be reused only by the place_opt call of the successive run. An RDE model dumped after clock_opt may be used in both the place_opt and clock_opt stages of the new run (the model need only be read in one time, before either place_opt or clock_opt). The "read_rde" command includes some checking as to whether there have been floorplan changes and, if so, may not allow the RDE model to be read in. In this case, the user should run the default flow and regenerate new RDE model information after the RDE capture stages if desired.

USAGE

run 1 : place_opt -from final_opto write_rde place_opt_RDE ... clock_opt -from final_opto write_rde clock_opt_RDE

run 2 : read_rde place_opt_RDE OR read_rde clock_opt_RDE place_opt -from final_opto ... read_rde clock_opt_RDE clock_opt

-from final_opto

EXAMPLES

The following example reads a RDE model for the current design.

```
prompt> read_rde -help
Usage: read_rde    # a RDE stream in command
      [file_name]          (filename to read in)
prompt> read_rde my.rde
```

SEE ALSO

write_rde(2)

read_saif

Reads a Switching Activity Interchange Format (SAIF) file and annotates switching activity information on nets, pins and ports in the current design.

SYNTAX

status **read_saif**
file_list
[-strip_path *source_name*]
[-path *target_name*]
[-scenarios *scenario_list*]
[-modes *mode_list*]
[-corners *corner_list*]
[-exclude_sdpd]
[-normalize true | false]
[-nocase]
[-report]
[-use_highest_toggle_rate]
[-ignore_name_mapping]

Data Types

file_list list
source_name string
target_name string
scenario_list list
mode_list list
corner_list list

ARGUMENTS

file_list

Specifies a list of SAIF files to be read. Each list item consists of the SAIF file name and optionally the weight, scaling ratio, strip path and the path. The syntax for each list item is

file_name
[-weight *weight_value*]
[-scaling_ratio *scaling_ratio_value*]
[-strip_path *source_name*]
[-path *target_name*]

The default for the **-weight** and **-scaling_ratio** options is 1.0 and when specified should be larger than 0.0. The **-strip_path** and **-path** options can be specified either individually for each file or, if multiple files have the same strip path or path, once for all files.

-strip_path *source_name*

Specifies the name of the instance of the current design as it appears in the SAIF file. The current design appears as an instance in the SAIF file. The read_saif command annotates all subinstances in the hierarchy of the specified instance, and annotates the instance itself. Each instance name that is specified must be complete, but without any trailing hierarchy separator (/).

-path *target_name*

Specifies the target instance on which the switching activity, in the SAIF file, is to be annotated. If this option not specified, the current instance is assumed to be the target instance.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Activity annotation is done separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, annotation is done for the current scenario in the design.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering occurs on modes.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If this option is not specified, no filtering occurs on corners.

-exclude_sdpd

When specified, the state-dependent and path-dependent (SDPD) switching activities are ignored while reading saif file(s). By default, the SDPD activities are read from saif file(s).

-normalize true | false

Controls whether the weight associated with each SAIF file is normalized, which means that the sum of the weighted activity values is divided by the sum of the weights.

By default (**true**), the command normalizes the weights.

If **false**, the command uses the weights as specified.

-nocase

When specified, the SAIF file is matched without considering the name case of the objects specified. The object matching is done in name-case insensitive manner.

-report

When specified, the command generates a summary which indicates number of objects present in saif file which were found in the design and also number of objects present in design which were found in the saif file. If this option is not specified, command summary is not generated.

-use_highest_toggle_rate

When this option is used, the nonSDPD activity with the highest toggle rate is selected from the list of specified SAIF files. When this option is specified, SDPD activities are ignored. This option cannot be used with the option *normalize* and *weight*.

-ignore_name_mapping

This option should be used when reading a SAIF generated from the current design where no name mapping needs to be performed, for example when reading a gate level SAIF file generated from the same netlist. This allows the existing name mapping database to be preserved but not used for the read_saif command.

DESCRIPTION

The **read_saif** command reads a list of SAIF files and annotates the switching activity attributes *rise_toggle_rate*, *fall_toggle_rate* and *static_probability* for nets, ports, and pins of the current design. If a particular object in the SAIF file cannot be found in the current design, the object is ignored and a warning message is provided. The read_saif command returns 1 if at least one of the objects in the file is successfully annotated. Otherwise, it returns 0. This command resets the switching activity for the design before reading the specified SAIF files. For the merging of SAIF files, one can read multiple SAIF files. The probabilities are multiplied with the weight factor and accumulated overall SAIF files. The toggle rates are multiplied with the weight factor and the scaling factor and accumulated over all SAIF files. The resulting probability and toggle rate is normalized (divided by the sum of weights), except when option -normalize false is used.

Whenever the '-nocase' option is used, the SAIF file is interpreted ignoring the name cases of the objects specified in SAIF file. An object with name 'ABC' in SAIF can be matched to object 'abc' in the netlist and vice-versa.

Saif information is saved with a block unless app option 'power.enable_activity_persistency' is set to 'off'.

Multicorner-Multimode Support

EXAMPLES

The following example annotates switching activity for the current block.

```
prompt> read_saif test.saif
```

The following example annotates switching activity for the current block that has been annotated as Test/U1 in simulation testbench.

```
prompt> read_saif test.saif -strip_path Test/U1
```

The following example annotates switching activity from the SAIF file for the instance Mult/U12 in the current block.

```
prompt> read_saif test.saif -path Mult/U12
```

The following example annotates the switching activity from a SAIF file and then reports it using the **get_switching_activity** command. The toggle rate values and static probability are derived from the SAIF file and the activity type is set to 'simulated'.

```
prompt> read_saif test1.saif
prompt> get_switching_activity [get_pins GPRs/iso]
(mode = default, corner = default, probability = 0.504926,
rise_toggle_rate = 9.61391e+06, fall_toggle_rate = 9.61391e+06,
type = simulated)
```

The following example merges two SAIF files test1.saif and test2.saif. The probabilities for the annotated cells will be (1 * probability(test1.saif) + 2 * probability(test2.saif)) / 3, and the toggle rates will be ((1 * 4 * toggle_rate(test1.saif) + 2 * 3 toggle_rate(test2.saif)) / 3. Since individual strip paths have not been specified per file, strip path for both will be ChipTop. Path for test1.saif will be InstDecode, and for test2.saif it will be GPRs.

```
prompt> read_saif {{test1.saif -scaling_ratio 4 -path InstDecode} \
  {test2.saif -weight 2.0 -scaling_ratio 3.0}} \
  -strip_path ChipTop -path GPRs
```

The following example merges two SAIF files test1.saif and test2.saif under the option use_highest_toggle_rate. The non-SDPD toggle rate and probabilities for the annotated cells will be chosen to be those of the one with highest toggle rate amongst test1.saif

and test2.saif . When this option is specified, SDPD activities are ignored.

prompt> **read_saif -use_highest_toggle_rate {{test1.saif} {test2.saif}}**

SEE ALSO

get_switching_activity(2)
report_power(2)
report_switching_activity(2)
reset_switching_activity(2)
set_switching_activity(2)
power.enable_activity_persistency(3)

read_sdc

Reads in a script in Synopsys Design Constraints (SDC) format.

SYNTAX

status **read_sdc**
file_name
[-echo]
[-syntax_only]
[-version *sdc_version*]

Data Types

file_name string
sdc_version string

ARGUMENTS

file_name

Specifies the name of the file that contains the SDC script to read.

-echo

Echoes each constraint as it is executed.

-syntax_only

Checks only the syntax and semantics of the SDC script.

-version *sdc_version*

Specifies the version of SDC to which the file conforms.

Allowed values are 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, and **latest** (the default).

DESCRIPTION

The **read_sdc** command reads in a script file in Synopsys Design Constraints (SDC) format. SDC is also licensed by external vendors through the Tap-in program.

By default, the **read_sdc** command executes the constraints as they are read. If there are errors in the script, it is possible that some constraints are applied, and some are not. You can use the **-syntax_only** option to check the script without applying any constraints. This also verifies conformance to the specified SDC version. If there are any errors during the checking phase, the result of the **read_sdc** command is 0.

Like the **source** command, the **read_sdc** command is sensitive to variables that control script execution, such as **sh_continue_on_error** and **sh_script_stop_severity**. The **read_sdc** command uses the **sh_source_uses_search_path** variable to determine if the **search_path** variable is used to find the script.

The **read_sdc** command supports several file formats. The file can be a simple ASCII script file, an ASCII script file compressed with gzip, or a Tcl bytecode script, created by TclPro Compiler.

Note that SDC does not allow command abbreviation. Also, the **exit** and **quit** commands do not cause the application to exit, but they do cause the reading of the SDC file to stop.

The latest SDC version supports the following commands. Those added for versions 1.3 and later are noted. Th list does not include some constraints that are ignored by the tool.

General Purpose Commands:

expr
list
set

Object Access Functions:

all_clocks
all_inputs
all_outputs
all_registers (1.7)
current_design
current_instance
get_cells
get_clocks
get_libs
get_lib_cells
get_lib_pins
get_nets
get_pins
get_ports
set_hierarchy_separator

Basic Timing Assertions:

create_clock
create_generated_clock (1.3)
group_path (1.7)
set_clock_gating_check
set_clock_groups (1.7)
set_clock_latency
set_clock_sense (1.7)
set_clock_transition
set_clock_uncertainty
set_data_check (1.4)
set_false_path
set_ideal_latency (1.7)
set_ideal_transition (1.7)
set_input_delay
set_max_delay

set_min_delay
set_multicycle_path
set_output_delay
set_propagated_clock

New options to existing supported commands:

create_clock -add (1.4)
create_generated_clock -add
create_generated_clock -master_clock
create_generated_clock -combinational (1.7)

Secondary Assertions:

set_disable_timing
set_max_time_borrow
set_timing_derate (1.5)

Environment Assertions:

create_voltage_area
set_case_analysis
set_drive
set_driving_cell
set_fanout_load
set_input_transition
set_ideal_network (1.7)
set_level_shifter_strategy
set_level_shifter_threshold
set_load
set_logic_dc
set_logic_one
set_logic_zero
set_max_area
set_max_capacitance
set_max_dynamic_power
set_max_leakage_power
set_max_transition
set_min_capacitance
set_min_fanout
set_min_porosity
set_operating_conditions
set_port_fanout_number
set_voltage (1.8)
set_wire_load_min_block_size
set_wire_load_mode
set_wire_load_model
set_wire_load_selection_group

For a complete guide to using SDC with Synopsys tools, see the *Using the Synopsys Design Constraints Format Application Note*, which is available in SolvNet at https://solvnet.synopsys.com.

The usage of some of the supported commands is restricted when reading SDC. In some cases, some options are not allowed. In some tools, some of the commands, such as the **set_logic*** commands, are not supported. If the SDC file contains a command that is not supported by the tool, the **read_sdc** command ignores the command and issues a warning message (for an example, see the EXAMPLES section). If the SDC file contains a command or option that is not supported by the SDC format, the **read_sdc** command issues an error message.

Multicorner-Multimode Support

This command works only on the current scenario.

EXAMPLES

The following example reads the SDC script named top.sdc.

```
prompt> read_sdc top.sdc -version 1.2

Reading SDC version 1.2...
1
```

The following example reads the SDC script named top2.sdc. The script contains commands not supported by SDC, so the **read_sdc** command issues CMD-005 messages for those commands.

```
prompt> read_sdc -syntax_only top2.sdc -version 1.2

Checking syntax/semantics using SDC version 1.2...

Error: Unknown command 'link_design' (CMD-005)

** Completed syntax/semantic check with 1 error(s) **
0
```

The following example shows the result when the SDC file contains a command that is not supported by the tool. One SDC message is generated per instance of an unsupported command. At the end of the read, the command generates a summary of all unsupported commands in the file. The SDC file is read into the tool.

```
prompt> read_sdc t2.sdc -version 1.2

Reading SDC version 1.2...
Warning: Constraint 'set_logic_zero' is not supported by the tool. (SDC-3)
Warning: Constraint 'set_logic_zero' is not supported by the tool. (SDC-3)
Warning: Constraint 'set_logic_one' is not supported by the tool. (SDC-3)

Summary of unsupported constraints:
Information: Ignored 1 unsupported 'set_logic_one' constraint. (SDC-4)
Information: Ignored 2 unsupported 'set_logic_zero' constraints. (SDC-4)
1
```

The following example shows the output generated by the **read_sdc** command when reading an SDC file with unsupported options. In an SDC file, the **current_design** can be used only to get the current design; no arguments are allowed.

```
prompt> read_sdc t3.tcl -echo -version 1.2
Reading SDC version 1.2...
current_design TOP
Error: extra positional option 'TOP' (CMD-012)
0
```

SEE ALSO

source(2)
write_sdc(2)
search_path(3)
sh_continue_on_error(3)
sh_script_stop_severity(3)
sh_source_uses_search_path(3)

read_signal_em_constraints

Reads the signal electromigration rules.

SYNTAX

status **read_signal_em_constraints**
[-encrypted]
[-format ITF | ALF]
file_name

Data Types

file_name string

ARGUMENTS

-encrypted

Indicates that the electromigration file is encrypted. This option is exclusive with the option *-format*.

-format ITF | ALF

Specifies the electromigration file format. The supported formats are **ITF** and **ALF**, and if this option is not specified, the data will be read as **ITF** format by default. This option is exclusive with the option *-encrypted*.

file_name

Specifies the ITF file from which to import signal electromigration data into the current design library. The specified ITF file should contain only signal electromigration information.

DESCRIPTION

The **read_signal_em_constraints** command reads signal electromigration data into the current design library. If the design library already contains signal electromigration information, it is replaced by the data in the new ITF file.

The command returns 1 if the file is successfully read in; otherwise, it returns 0.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The first example reads a signal electromigration file named test.itf into the current design library. The second example reads a encrypted signal electromigration file named test.itf.enc into the current design library. The third example reads a signal electromigration file named test.alf into the current design library.

```
prompt> read_signal_em_constraints test.itf
prompt> read_signal_em_constraints -encrypted test.itf.enc
prompt> read_signal_em_constraints -format ALF test.alf
```

SEE ALSO

current_lib(2)
read_tech_file(2)
search_path(3)

read_tech_file

Reads a technology file into the current library.

SYNTAX

int **read_tech_file**
[-convert_sites *convert_sites*]
file_name

Data Types

file_name Technology file to read
convert_sites List of technology file sites to convert

ARGUMENTS

-convert_sites *convert_sites*

A list of technology file sites to convert. This option can be used only with the **-technology** option.

file_name

The technology file you want to read.

RETURN VALUE

Returns 1 if the technology file is successfully read in, and 0 if it is not.

DESCRIPTION

The **read_tech_file** command reads a technology section into the current library. If sites are defined in the technology file, the -convert_sites option can be used to map them from one name to another. The same option exists in the read_def command to make the site names match among all the input data.

If the new technology file is missing layers defined in the existing technology data, or if some layers have different names/numbers, the **read_tech_file** command will allow the replacement. In these cases, the tool issues information messages and existing objects on the missing layers become unbound objects. When layers are re-created or read in through another technology file replacement, these objects are rebound to the layers.

EXAMPLES

The following example read in a technology file "tcb90g.tf" into the current library.

```
prompt> read_tech_file tcbn90g.tf
Information: technology file '/user/joe/work/tcbn90g.tf' is loaded
{"r4000"}
```

SEE ALSO

create_lib(2)
write_tech_file(2)
report_unbound(2)
search_path(3)

read_tech_lef

Reads one or more LEF (Library Exchange Format) technology data files.

SYNTAX

string **read_tech_lef**
[-design *design*]
[-merge_action add | ignore | overwrite | update]
[-syntax_only]
file_names

Data Types

design	collection
file_names	list

ARGUMENTS

-design *design*

Specifies the block in which to create the LEF technology data. If this is not specified, LEF technology data will be created in the current block.

-merge_action add | ignore | overwrite | update

Specifies the strategy for merging LEF data into an existing block. The strategy is used only when there has a matched object found in existing block. The definition for each option value is:

- **add** (the default) : Adds a new object to existing block by making name unique
- **ignore** : Skips the new object
- **overwrite** : Deletes the existing object in block, and then create the new object.
- **update** : Combines the new object with the existing object in block, retaining both data sets. For objects that do not have components, this will be same as overwrite. For objects that have components, the merge depends on the components. For components that should be unique or already exists, the command overwrites them. For components that do not have name or do not exist yet, the command adds them.

-syntax_only

Specifies that the LEF files are processed only to check the syntax and semantics of the files.

file_names

Specifies names of one or more LEF files to be read.

DESCRIPTION

The command reads LEF technology data into an existing block.

To locate files that have relative path names, the command uses the **search_path** variable and searches for each file in each directory listed in the **search_path**. Files that have absolute path names are loaded as though there were no **search_path**. To determine the file that **read_tech_lef** loads, use the **which** command.

The **-merge_action** option customize how LEF technology data is merged into the existing block.

This command is only available in **icc2_shell**.

EXAMPLES

The following example reads the file via_rule.lef file.

```
prompt> read_tech_lef via_rule.lef
1
```

The following example reads via_rule.lef into an existing block called my_block and ignores an object if it already exists in the block.

```
prompt> read_tech_lef via_rule.lef -design my_block -merge_action ignore
```

SEE ALSO

read_lef(2)
search_path(3)

read_test_model

Reads a test model file.

SYNTAX

status **read_test_model**
[-format ctl | icl | pdl | ict | wks | hdl]
[-design *design_name*]
filename

Data Types

design_name string
filename string

ARGUMENTS

-format ctl | **icl** | **pdl** | **ict** | **wks** | **hdl**

Specifies the format of the test model.

By default, the model is assumed to be in .ctl format.

-design *design_name*

Specifies the name of the design to which to attach the test model. This is meaningful only for CTL test models. By default, the model is attached to the current design.

model_files

Specifies the names of the test model files to read. This argument is required. When you specify the format as **ctl**, you can specify only one file.

DESCRIPTION

The **read_test_model** command reads test model files.

The test model files represent the test behavior of designs. The attributes that are represented include all DFT signals and their purpose, the test timing, and the sequence of events needed to shift test vectors in and out of the design. The /fBread_test_model/fP command reads in only the test model and the interface definition of a design, and ignores any other information. Since the gate-level implementation of a design is not needed, significant memory and runtime savings can be achieved.

A successful read of a model overwrites any test model information in memory.

Note that for standard IEEE1838, TAP port specifications inside Internal statment of testmode should adhere the following points.

- **Primary Tap:** User data should contain both "IEEE1149.1" and "IEEE1838" standards to define the port as Primary TAP. Last user data is to define the name of the tap set, it is optional for PTAP ports.

  For example, "tdo" { DataType ScanDataOut User "IEEE1149.1" User "IEEE1838" User "TAP set name" { ScanDataType Internal; } }

- **Secondary Tap:** User data should contain only "IEEE1838" standard to define the port as secondary TAP. Last user data is to define the name of the die to which the port belongs, if there are multibple STAP die's available, it is mandatory to provide the state name.

  For example, "myStapTdo" { DataType ScanDataOut User "IEEE1838" User "STAP set name" { ScanDataType Internal; } }

Test models are stored on disk using the **write_test_model** command.

EXAMPLES

```
prompt> current_design
{des_unit.nlib:des_unit.design}

prompt> read_test_model myModel.ctl

File name: myModel.ctl

Warning: CTL Model read for design des_unit (DFT-2000)
```

SEE ALSO

write_test_model(2)

read_test_protocol

Reads a STIL test protocol file into memory.

SYNTAX

status **read_test_protocol**
-section *section_name*
[-test_mode *test_mode_name*]
file_name

Data Types

section_name string
test_mode_name string
file_name string

ARGUMENTS

-test_mode *test_mode_name*

Specifies a test mode name to read and store a test protocol. If not specified, the current test mode is used by default. This option is only supported when updating the post-DFT test protocol.

-section *section_name*

Specifies a section name of the protocol. With this option, the **read_test_protocol** command reads only the specified section and ignores others. The only valid parameter value is **test_setup**, which describes the initialization sequence.

input_file_name

Specifies the name of the test protocol file in STIL format to read.

DESCRIPTION

This command reads a test protocol file into memory. Reading a test protocol file defines the scan test protocol for the current design.

The scan test protocol provides a means to formally define the process by which a design is tested, such as the sequence of scan-in vectors, parallel vectors, and scan-out vectors performed for each test pattern. The protocol defined for a design is used to drive scan-test design rule checking.

The format of the test protocol file is STIL. See the TestMAX DFT and TestMAX ATPG user guides for details of the protocol file

format.

The **read_test_protocol** command enables you to define an arbitrary test protocol for a design by reading in a text file describing the protocol. Use **write_test_protocol** to write out an existing protocol for a design in the same text format.

Normally, this command removes any protocol that may be present in memory through the previous use of the **read_test_protocol** or **create_test_protocol** commands.

When the **read_test_protocol -section test_setup** command is executed before pre-DFT DRC has been performed, the **test_setup** section from the specified protocol file is read in and used in place of the default setup protocol created by the tool. The provided **test_setup** section must include all preconditioned signals defined by the **set_dft_signal** command, such as clock, set, reset, and constant signals. Any STIL **Loop** constructs are retained as-is.

To modify the default setup protocol used by pre-DFT DRC, you can write out the default test protocol, modify the **test_setup** section, then read it back in; see the EXAMPLES section for more information.

When the **read_test_protocol -section test_setup** command is executed after DFT insertion has been performed, the **test_setup** section of the in-memory test protocol created by DFT Compiler is merged with the **test_setup** section from the specified protocol file. This merged in-memory protocol is used for any subsequent DRC. Any STIL **Loop** constructs are unrolled during this merging process. If multiple test modes were created by DFT Compiler, you can update each one using the **-test_mode** option of this command, or the **current_test_mode** command.

With the STIL protocol format, you cannot distinguish between a clock signal and an asynchronous signal. You must define asynchronous signals with the **set_dft_signal** command before reading your custom STIL protocol.

EXAMPLES

The following example reads a test protocol from the **newUPC1.spf** file:

```
prompt> read_test_protocol newUPC1.spf
Reading test protocol file 'newUPC1.spf'...
```

The following example reads a test protocol for the **UPC1** design from the **UPC1.spf** default file:

```
prompt> read_test_protocol
Reading test protocol file 'UPC1.spf'...
```

The following example defines the asynchronous signal **rstn** for design UPC1 and reads a test protocol from the file UPC1.spf:

```
prompt> set_dft_signal -type Reset -active_state 0 -port rstn

prompt> read_test_protocol UPC1.spf
```

The following example reads a **test_setup** section of the protocol. For example, assume that the **init_sequence.spf** file has the following **test_setup** vector sequence:

```
MacroDefs {
  "test_setup" {
    W "_default_WFT_";
    V { "A" = 0; "B" = 0; "CP" = 0; }
    V { "A" = 1; "B" = 1; "CP" = P; }
    V { "A" = 0; "B" = 0; "CP" = 1; "AA" = 1; "BB" = 1; }
  }
}
```

The following command reads in the initialization sequence and ignores the rest of the protocol file:

```
prompt> read_test_protocol -section test_setup init_sequence.spf
```

The following example creates a default test protocol, modifies it, and reads the modified **test_setup** back in:

```
prompt> create_test_protocol

prompt> write_test_protocol -output test_setup_mod.spf

  (manually modify test_setup in file,
  keeping pre-conditioned signals as needed)

prompt> read_test_protocol -section test_setup test_setup_mod.spf
```

The following example updates the post-DFT test setup protocol for two test modes:

```
prompt> insert_dft

prompt> read_test_protocol -section test_setup \
        -test_mode Internal_scan postDFT.spf

prompt> read_test_protocol -section test_setup \
        -test_mode ScanCompression_mode postDFT.spf

prompt> current_test_mode Internal_scan

prompt> dft_drc

prompt> current_test_mode ScanCompression_mode

prompt> dft_drc
```

SEE ALSO

create_test_protocol(2)
dft_drc(2)
set_dft_signal(2)
write_test_protocol(2)

read_verilog

Reads in one or more Verilog files.

SYNTAX

string **read_verilog**
[-library *library_name*]
[-design *design_name*]
[-top *module_name*]
file_names

Data Types

library_name string
design_name string
module_name string
file_names string

ARGUMENTS

-library *library_name*

Specifies the name of the library to contain the design. If the current library is not set, the command creates a library using the first input file name. By default, the command saves the design into the current library. If the reference library list is not set, then issues a warning.

-design *design_name*

Specifies the name of the design and label to read in, in designName[/labelName] format. If design name is not specified, the top module is used as the design name. If label name is not specified, the default label, specified with the file.verilog.default_user_label app_option, will be used. Reading into an opened design is not allowed. The design must not already exist or be closed.

-top *module_name*

Specifies the name of the top module in the design hierarchy. By default, the top module is the module that is not instantiated by any other modules.

-as_block

Specifies that a new block should be created to store the read modules.

file_names

Specifies the names of one or more input Verilog netlist files.

DESCRIPTION

Reads one or more structural, gate-level Verilog netlists into the tool. To locate Verilog files that have relative pathnames, the command uses the **search_path** variable and searches for each file in each directory listed in the **search_path**. Files with absolute pathnames are loaded independent of the **search_path** variable. Use the **which** command to determine the file that **read_verilog** loads. After reading the Verilog files with the **read_verilog** command, use the **list_blocks** command to view the design objects. To remove designs, use the **remove_block** command.

The command saves the read modules into the current block if there is an open current block defined, and the block is not yet linked. This behavior can be disabled so a new block is created with the -as_block argument, or if -design is used.

When targeting an existing block a redefinition of a module will overwrite an existing definition.

The Verilog netlists must contain fully-mapped, structural designs. The tool cannot link or perform timing analysis with netlists that are not fully mapped at the gate level. There must be no high-level Verilog constructs in the netlist.

The Verilog file can be a simple ASCII file or a compressed gzip file. Each gzip file is uncompressed to a temporary file in the directory defined by the **icc2_tmp_dir** variable.

Generally, the Verilog reader does not create unconnected nets and discards unconnected instances. To keep unconnected nets in your design, set the **file.verilog.keep_unconnected_nets** app option to **true**. Also, if a module has one or more references to a reference name, but no instances have connections, those instances are omitted if the **file.verilog.keep_unconnected_cells** app option is set to "false" (the default). Discarding unconnected instances saves memory for instances of filler cells and other cells with no pins.

Additionally, **read_verilog** command prints summary at the end of run. The following describes each field:

- Number of modules read: <no. of modules read from verilog file>
- Top level ports: <no. of ports in top module including PG ports>
- Total ports in all modules: <total no. of ports in all the modules including PG ports>
- Total nets in all modules: <total no. of nets in all the modules including PG nets>
- Total instances in all modules : <total no. of instances in all the modules>
- Elapsed = <clock time>, CPU = <cpu time>

Limitations of the Verilog Reader

The Verilog reader has the following limitations:

- No non-structural constructs are supported. For details, see the "Supported Language Subset for the Verilog Reader".
- Parameters are not supported. Parameter and defparam statements are read and ignored.
- Gate instantiations are not supported.

Supported Language Subset for the Verilog Reader

The Verilog reader supports a small structural subset of the Verilog language. Additional constructs are recognized and ignored. Supported constructs are as follows:

- module/endmodule
- input, inout, and output

- wire
- tri, wor, and wand: These constructs are supported and are treated the same as the wire construct. That is, there is no special significance to these constructs.
- supply0 and supply1: These constructs create wires that are logic 0 and 1, respectively.
- assign
- tran: An exception from the gate instantiation subset, tran is supported to the extent that it relates one wire to another, as with assign. Beyond that, tran has no special significance.
- The preprocessor directives `define, `undef, `include, `ifdef, `else, and `endif are not supported by default. The **read_verilog** command targets structural, machine-generated Verilog, which rarely contains these constructs. Set the **file.verilog.enable_vpp** app option to true to enable a preprocessor phase that expands these constructs and outputs a temporary file to the directory defined by the **icc2_tmp_dir** variable.
- All simulation directives (for example, `timescale) are recognized and ignored.
- The specify/endspecify construct and its contents are recognized and ignored.
- Like HDL Compiler, the comment directives *translate_off* and *translate_on* are used to bypass Verilog features not supported by the reader. The following example shows how to bypass an unsupported feature:

```
// synopsys translate_off
nand (n2, a1, s2);
/* synopsys translate_on */
```

EXAMPLES

The following example defines a search path and reads the newcpu.v Verilog file.

```
prompt> set search_path "/designs/newcpu/v1.6 /libs/cmos"
/designs/newcpu/v1.6 /libs/cmos

prompt> read_verilog newcpu.v
Loading verilog file '/designs/newcpu/v1.6/newcpu.v'
Number of modules read: 6
Top level ports: 124
Total ports in all modules: 514
Total nets in all modules: 598
Total instances in all modules: 18
Elapsed = 00:00:00.00, CPU = 00:00:00.00
1
```

SEE ALSO

remove_blocks(2)
which(2)
bus_naming_style(3)
search_path(3)
shell.tmp_dir_path(3)

verilog_options(3)

read_verilog_outline

Reads one or more Verilog files to create an outline design.

SYNTAX

string **read_verilog_outline**
[-library *library_name*]
[-design *design_name*]
[-top *module_name*]

[-partition | -allocation]
[-target_block_size *target_block_size*]
[-glue_cell_count *glue_cell_count*]
[-target_cell_count *target_cell_count*]
[-large_threshold *cell_count_threshold*]

[-depth *hierarchy_depth*]
[-keep_port_depth *port_depth*]
[-dense_modules *dense_module_names*]
[-port_modules *port_module_names*]
[-sparse_modules *sparse_module_names*]

[-leaf_cells *leaf_cell_names*]
[-buffer_cells *buffer_cell_names*]
[-macro_cells *macro_cell_names*]

file_names

Data Types

library_name string
design_name string
module_name string
target_block_size integer
glue_cell_count integer
target_cell_count integer
cell_count_threshold integer
hierarchy_depth integer
port_depth integer
dense_module_names list
port_module_names list
sparse_module_names list
leaf_cell_names list
buffer_cell_names list
macro_cell_names list
file_names list

ARGUMENTS

-library *library_name*

Specifies the name of the library to contain the design. If the current library is not set, the command creates a library using the first input file name. By default, the command saves the design into the current library. If the reference library list is not set, then issues a warning.

-design *design_name*

Specifies the name of the design and label to read in, in designName[/labelName] format. If design name is not specified, the top module is used as the design name. If label name is not specified, the default label, specified with the file.verilog.default_user_label app option, is used. Reading into an opened design is not allowed. The design must not already exist or be closed.

-top *module_name*

Specifies the name of the top module in the design hierarchy. By default, the top module is the module that is not instantiated by any other modules.

-partition

Use the partition method to select dense modules.

-allocation

Use the allocation method to select dense modules.

-target_block_size *target_block_size*

Specifies the target for the number of cells in each partition block.

-glue_cell_count *glue_cell_count*

Specifies the target for the number of leaf cells to allocate for glue modules of the partition blocks.

-target_cell_count *target_cell_count*

Specifies the target for the total number of leaf cells to create.

-large_threshold *cell_count_threshold*

Specifies the number of leaf cells that a module can contain before it is considered to be a large module.

-depth *hierarchy_depth*

Specifies the number of top levels of module hierarchy to make dense.

-keep_port_depth *port_depth*

Specifies the number of levels of module hierarchy to preserve port interfaces.

-dense_modules *dense_module_names*

Specifies a list of module names to have dense representations.

-port_modules *port_module_names*

Specifies a list of module names to have port interfaces preserved.

-sparse_modules *sparse_module_names*

Specifies a list of module names to have sparse representations.

-leaf_cells *leaf_cell_names*

Specifies a list of cell reference names to be treated as leaf cells.

-buffer_cells *buffer_cell_names*

Specifies a list of cell reference names to be treated as buffer cells.

-macro_cells *macro_cell_names*

Specifies a list of cell reference names to be treated as macro cells.

file_names

The names of one or more Verilog files to be read.

DESCRIPTION

Reads one or more structural, gate-level Verilog netlists into the tool as an outline design. An outline design is used for high capacity design planning. The outline design contains hierarchy, but no leaf cells or nets. An outline design cannot be used for optimization or timing.

After committing the design block hierarchy with the **commit_block** command, use the **expand_outline** command to expand each outline representation into a full design based on the original Verilog. It is important to keep the original Verilog files untouched in their original locations before you run the **expand_outline** command.

To enable Data Flow Analysis, top-level outline modules should contain dense netlist connectivity. Lower-level outline modules that don't need that information should contain sparse netlist connectivity. Use the **read_verilog_outline** command options to specify which modules should have a dense or sparse representation.

One of two methods are used to determine which modules are marked as dense or sparse. Use the partition method to create individual blocks of a specified hierarchical size. Use the allocation method to explore as much of the design hierarchy as possible within a memory budget of a maximum number of leaf cell instances. The default is the partition method. This can be changed with the *plan.outline.partition* app option.

For the partition method, top-level modules in the module hierarchy are marked as dense until the hierarchical cell count beneath a module is less than a target block size. Buffers and inverters are included in the hierarchical cell count. This maintains connectivity in the top levels of hierarchy which enables Data Flow Analysis. The target block size is specified by the *plan.outline.target_block_size* app option, which defaults to 1,000,000 leaf cells. Use the **-target_block_size** option to override the default setting.

In the partition method, after the partition blocks are designated, small module hierarchies within each partition are made dense. This maintains connectivity between the partition blocks for better Data Flow Analysis. The target number of leaf cells to allocate to these module hierarchies is specified by the *plan.outline.glue_cell_count* app option, which defaults to 100,000 leaf cells. A value of -1 skips marking any glue modules as dense. Buffers and inverters are not represented in the outline design and are not included in the target glue cell count. Use the **-glue_cell_count** option to override the default setting.

For the allocation method, top-level modules in the module hierarchy are marked as dense until the target leaf cell count is reached. This count allows a rough trade-off between memory used for the outline design and the level of detail represented in the design. Buffers and inverters are not represented in the outline design and are not included in the target leaf cell count. The target leaf cell count is specified by the *plan.outline.target_cell_count* app option, which defaults to 1,000,000 leaf cells. Use the **-target_cell_count** option to override the default setting.

In the allocation method, modules that meet a leaf cell count threshold are considered large modules that should be represented as sparse modules in an outline design. Buffers and inverters are not represented in the outline design and are not included in the leaf cell count. The leaf cell count threshold is specified by the *plan.outline.large_threshold* app option, which defaults to 10,000 leaf cells. Use the **-large_threshold** option to override the default setting.

For Data Flow Analysis, dense connectivity is needed in top-level modules. The module hierarchy depth that should be marked as dense is specified by the *plan.outline.dense_module_depth* app option, which defaults to 1. Use the **-depth** option to override the default setting. A depth value of 1 forces the top module to be dense, a depth value of 2 forces the top module and its immediate children in the module hierarchy to be dense, and so on. A value of 0 has no effect. A value of -1 marks all non-sparse modules as dense. The hierarchy depth overrides the target cell count specified by the **-target_cell_count** option.

Modules and their parent modules in the module hierarchy up to the top module can be treated as dense modules by specifying their module names in the **-dense_modules** option list. The dense modules list overrides the target cell count specified by the **-target_cell_count** option.

Modules can be treated as sparse modules by specifying their module names in the **-sparse_modules** option list. Modules specified as sparse override all other options.

Dense modules have complete port interfaces, sparse modules do not. Use the *plan.outline.keep_port_depth* app option to specify the module hierarchy depth that should retain complete port interfaces. The default is 1. That value can be overridden with the **-keep_port_depth** option. A value of 0 or 1 retains the port interface of the top module. A value of 2 retains the port interface of the top module and its immediate children, and so on. A value of -1 retains the port interfaces for all modules.

Modules can be explicitly marked to retain their port interfaces by specifying their module names in the **-port_modules** option list.

The library's reference library list is examined to determine what cells are leaf cells. These cells are not instantiated into sparse modules, but they are counted so the area of each module can be computed. Buffer and inverter cells are not instantiated into sparse or dense modules, they are always counted so the area of each module can be computed. Large macro cells are important for design planning, so they are always instantiated.

The **-leaf_cells** option can be used to specify the reference names of additional cells that should be treated as leaf cells and not instantiated into the sparse modules.

The **-macro_cells** option can be used to specify the reference names of additional cells that should be treated as macro cells and are always instantiated into the outline design.

In dense modules, netlist connectivity is mostly preserved. Inverters and buffer cells from the reference library are treated as wires so connections through those cells can still be seen, minus the buffers and inverters themselves.

The **-buffer_cells** option can be used to specify the names of additional cells that should be treated as wires connecting nets.

Nets and leaf cells are not represented in sparse modules. Leaf cells are counted so the area of each module can be computed. Sparse modules contain port interfaces only if a parent of the module is a dense module. Since a module must have a port interface to be committed as a block, make sure that modules which are block candidates have their port interfaces preserved.

Nets are still represented for flip_chip_driver cells even in sparse modules. When the flip_chip_driver cell is sparse, the module and the modules that contain them (up to the top) will be marked to retain their port interfaces, so there are nets to make the connections up to the top-level ports.

All port interfaces and nets are restored in the full design by the **expand_outline** command. Otherwise the **read_verilog_outline** command is similar to the **read_verilog** command. See the **read_verilog** command documentation for more information on options and variable settings.

EXAMPLES

In the following example, the r4000.v.gz compressed Verilog file is read into an outline design. The alu is committed as a separate

block, and the top design is expanded.

```
prompt> set_ref_libs -ref_libs {"tcbn90ghvt" "tcbn90glvt"}
tcbn90ghvt tcbn90glvt
prompt> read_verilog_outline -top r4000 r4000.v.gz
Loading verilog file '/usr/.../r4000.v.gz'
1
prompt> current_design r4000
{"r4000"}
prompt> link
Information: units loaded from library 'tcbn90ghvt' (LNK-040)
Design 'r4000' was successfully linked.
1
prompt> commit_block alu
1
prompt> expand_outline
Loading verilog file './r4000.hcvr.v.gz'
1
prompt>
```

SEE ALSO

commit_block(2)
expand_outline(2)
read_verilog(2)
set_app_options(2)
set_ref_libs(2)
outline_attributes(3)
plan.outline_options(3)

read_virtual_pad_file

Creates virtual pads from a file.

SYNTAX

status **read_virtual_pad_file**
file_name *file_name*

Data Types

file_name string

ARGUMENTS

file_name *file_name*

Specifies the file that contains virtual pad net names and locations.

DESCRIPTION

This command reads a file that contains net and coordinate information for virtual pads, and instantiates the virtual pads into the design.

The virtual pad file uses the following format, where the power net name is followed by lines containing x-coordinate, y-coordinate, and layer name for each pad location. You can also specify "auto" as the layer name.

```
power_net_name_a
x1 y1 layer_name1
x2 y2 layer_name2

power_net_name_b
x3 y3 layer_name3
x4 y4 layer_name4
```

The following example assigns virtual pads for nets VDD and VSS.

```
VDD
1632.565  1183.195  auto
576.565   987.625   auto
1566.565  589.105   auto
675.565   1201.645  auto
```

```
1401.565  758.845   auto
VSS
1071.565  1365.565  auto
840.565   943.345   auto
708.565   1068.805  auto
1566.565  762.535   auto
```

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example creates virtual pads from a file named test.vpad

```
prompt> read_virtual_pad_file test.vpad
```

SEE ALSO

analyze_power_plan(2)
remove_virtual_pads(2)
report_virtual_pads(2)
set_virtual_pad(2)
write_virtual_pad_file(2)

rebind_block

Rebind references in a block.

This command is deprecated, please use link_block command.

SYNTAX

int **rebind_block** [-verbose] [*block*]

string *block*

ARGUMENTS

-verbose

The display verbose messages.

block

The block to be rebound; the default is the current block.

DESCRIPTION

Rebinds block references for the specified *block* or the current block using the library's ref_lib list.

Libraries specified in the ref_lib list are searched in order for references to physically hierarchical blocks and lib cells. The ref_lib list uses the *search_path* to locate libraries. The ref_lib list is usually set when the library is created with the **create_lib** command, but can be updated with the **set_ref_libs** command.

Unresolved Verilog references from **link_block** will be bound using the view switch list if the referenced block is now available in the ref_lib list.

The **rebind_block** command does not change the bound block view for each cell. To change the view referenced by a cell, use the **change_view** command.

Pins on each cell that reference the block are updated to match the ports on the referenced block. Pins can be added or deleted as needed. If a pin that is a candidate to be deleted has a net connection, the rebind fails for that cell. That block reference is left unbound and the cell is unchanged.

The operation of the **rebind_block** command is similar to the **link_design -rebind** command. The **rebind_block** command only rebinds one block, while the **link_design -rebind** command rebinds all blocks in the block hierarchy.

Unresolved References

Cells which fail to be rebound are left as unbound references. These references appear as leaf cells in the block. To resolve an unbound reference, update the library's ref_lib list or the *search_path* to include the library containing the desired referenced block, and rebind the block.

The number of unresolved references to have detailed error messages can be selected using the *link.reference_limit* app option. After this limit has been reached, a summary of the number of unresolved references is given. You can change this limit with the **set_app_options** command. Information on **rebind_block** app options is found on the *link_options* manual page.

Rebind Errors

Missing block references or incompatible port references will cause **rebind_block** to fail. To correct these failures, update the ref_lib list or *search_path* and rebind the block.

EXAMPLES

In the following example, block TOP in library blockLib is linked to physically hierarchical blocks MID and BOT, and the lib cell library leafCellLib.

```
prompt> rebind_block -verbose TOP
Using libraries: designLib leafCellLib
Rebind: Collapsing hierarchy for cell 'u2/uMID'.
Rebind: Collapsing hierarchy for cell 'u1/u2/uMID'.
Rebind: Collapsing hierarchy for cell 'u1/uBOT'.
Rebind: Remapping pins on reference MID
Block 'TOP' was successfully rebound.
1
prompt> link_block -verbose
Using libraries: designLib leafCellLib
Visiting block tiny:TOP.design
Visiting block tiny:BOT.design
Visiting block tiny:MID.design
Expanding block tiny:TOP.design
Block 'TOP' was successfully linked.
1
prompt>
```

SEE ALSO

create_lib(2)
current_block(2)
current_design(2)
get_view_switch_list(2)
list_blocks(2)
link_block(2)
read_verilog(2)
remove_blocks(2)

set_app_options(2)
set_ref_libs(2)
set_view_switch_list(2)
link_options(3)
search_path(3)

reconnect_fishbone_style_power_switch

Performs blockage-aware fishbone style power switch reconnection.

SYNTAX

collection **reconnect_fishbone_style_power_switch**
-sw_cells *list_of_power_switch_names*
-ctrl_pin_name *library_control_pin_name*
-ack_pin_name *library_acknowledge_pin_name*

Data Types

list_of_power_switch_names list
library_control_pin_name string
library_acknowledge_pin_name string

ARGUMENTS

-sw_cells *list_of_power_switch_names*

Specifies power switch cell names which will be reconnected. The specified power switch cells must be connected by the **connect_power_switch -mode fishbone** command. This option is required.

-ctrl_pin_name *library_control_pin_name*

Specifies the control pin name for the power switch library cell. This option is required.

-ack_pin_name *library_acknowledge_pin_name*

Specifies the acknowledge pin name for the power switch library cell. This option is required.

DESCRIPTION

This command performs blockage-aware fishbone style reconnection for power switches which were connected by the **connect_power_switch -mode fishbone** command. This command detects long connections between power switches and reconnects the power switches to the nearest neighbors. This command will not reconnect the main trunk cells. It cannot handle multiple groups of power switches simultaneously and it cannot be called multiple times for the same group of power switches after **connect_power_switch -mode fishbone**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example reconnects the power switches. The control pin is SLEEP, the acknowledge pin is SLEEPOUT.

```
prompt> reconnect_fishbone_style_power_switch -sw_cells $sw_names \
  -ctrl_pin_name SLEEP -ack_pin_name SLEEPOUT
```

SEE ALSO

connect_power_switch(2)

record_layout_editing

This command records changes to the layout change for a restricted set of objects and writes the changes to a Tcl script file.

SYNTAX

status **record_layout_editing**
-start | -stop
[-output *file_name*]

Data Types

file_name string

ARGUMENTS

-start

Begins recording changes to the layout for the current design. Layout changes are recorded until you perform **record_layout_editing -stop** or close the design.

-stop

Stops recording of changes to the layout writes out the changes to the output file specified by the **-output** option. This option must be used together with the **-output** option.

-output *file_name*

Specifies the output Tcl file name into which to write the layout changes. The file contains changes from the point when **record_layout_editing -start** is run to current point. This option must be used together with the **-stop** option.

DESCRIPTION

This command records the layout changes and writes out the changes to a Tcl script file. You can source the Tcl script file to re-create the changes. It could help to realize the parallel layout editing of the design.

Run **record_layout_editing -start** to begin recording layout changes. When you run **record_layout_editing -stop -output**, layout changes since you ran **record_layout_editing -start** are written out as Tcl commands to the specified file. If you run **record_layout_editing -start** and the tool is already recording layout changes, the tool issues a warning message, discards the previous recording, and begins a new recording. If you close design while recording layout changes, the tool issues a warning message and discards the recording.

Parallel Layout Editing using record_layout_editing

You can use the **record_layout_editing** command to record layout changes on individual design blocks, then reapply the combined changes to a master design copy. The basic flow of the technique is as follows:

1. Design the partition.
2. Deploy each layout partition to separated designers
3. Record the layout changes for each design with the following steps:
   1) record_layout_editing **-start**
   2) Edit the layout
   3) record_layout_editing **-stop** -output change.tcl
4. Collect each designer's record of changes and source the changes to the original design one by one. Using this technique, the final design will contain all the layout changes.

Supported Object Types

Only changes for objects in the top block will be saved. The supported object types are as the following:

- Shape
- Via
- Terminal
- Via_def

Supported Editing Operations for Recording

The supported layout editing operation includes only the following:

- Tcl commands for creating layout objects
- Tcl commands for removing layout objects
- Set attributes by using **set_attribute**
- GUI move and resize

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example records a layout editing session to remove shape RECT_32_0.

```
prompt> record_layout_editing -start
prompt> remove_shapes RECT_32_0
prompt> record_layout_editing -stop -output change.tcl
```

SEE ALSO

set_attribute(2)
remove_shapes(2)

record_signoff_eco_changes

Records ECO changes for the incremental signoff ECO flow.

SYNTAX

int **record_signoff_eco_changes**
-start [-input *file_name*]

int **record_signoff_eco_changes**
-stop
[-def]
[-compress]

int **record_signoff_eco_changes**
-init
[-def]
[-compress]

Data Types

file_name PrimeTime ECO Tcl file

ARGUMENTS

-start

Start to record ECO changes if the recording is not started yet, or else it will error out. This option is mutually exclusive with the **-stop** and **-init** options.

-input *file_name*

Specify a Tcl file for signature check. This option works together with the **-start** option. The tool checks signoff ECO signature consistency between the Tcl file and the current block. If the signatures are inconsistent, or there is no signature inside design or Tcl file, the check fails and the command issues an error message and exits. If the check passed, this command will automatically source the Tcl file.
The Tcl file input is the PT-ECO Tcl file generated by PrimeTime ECO flow. The signature check provide one extra layer protection to make sure the PT-ECO Tcl file applied to the right block.

-stop

Terminate the recording if recording mode is on and save necessary data to disk. This option is mutually exclusive with the **-start** and **-init** options.

-init

Terminate the recording if recording mode is on, and save the full design data to disk as an initial point for the incremental signoff ECO flow. This option is mutually exclusive with the **-start** and **-stop** options.

-def

Save the DEF file. This option is used with the **-stop** or **-init** option.

-compress

Save the DEF and Verilog files in compressed format if available. This option works with the **-stop** or **-init** option.

DESCRIPTION

The **record_signoff_eco_changes** command records and saves the design ECO changes for the incremental signoff ECO flow. The command can record cell, netlist and physical geometry changes, and save the ECO changes to disk for the current block.

Inside the tool session for incremental signoff ECO flow, the timing ECO commands are not limited to PT-ECO Tcl. You can provide timing ECO commands manually to update the design. However, the logical recording will be terminated automatically if functional ECO commands are issued in the flow.

The saved ECO changes are used to benefit StarRC and PrimeTime runtime for the multiple iterations of incremental signoff ECO flow. It will always save an information file and full NDM db. Full DEF file will be saved if the *-def* option is available. If the logical recording is terminated, or option *-init* is used, full Verilog file will be saved. Depending on the recording status, other design change files may be saved also. These files will be used by StarRC and PrimeTime internally inside the incremental signoff ECO flow and are not be user-readable. If the current block is closed before the recording is stopped, all recorded information is discarded.

EXAMPLES

The following example is to create an initial point for incremental signoff ECO flow:

```
prompt> record_signoff_eco_changes -init
```

The following example starts to record the design ECO changes.

```
prompt> record_signoff_eco_changes -start -input pt_eco.tcl
```

The following example saves the recorded ECO changes.

```
prompt> record_signoff_eco_changes -stop
```

The following example is a simple incremental signoff ECO flow.

```
prompt> open_block LIB.ndm:BLOCK
prompt> record_signoff_eco_changes -start -input pt_eco.tcl
prompt> place_eco_cells
prompt> route_eco
prompt> record_signoff_eco_changes -stop
```

SEE ALSO

place_eco_cells(2)
route_eco(2)

recover_auto_save

Opens the auto-saved library of the current library.

SYNTAX

status **recover_auto_save**

DESCRIPTION

This command recovers and opens the auto-saved library of the current library. You can also recover and open the latest auto-saved library through the command option -recover of open_lib. The auto-saved library is recovered, opened and merged into the current or original library.

EXAMPLES

The following example recovers the auto-saved library to the current library.

```
prompt> report_auto_save
*********************************************************
Original Library : r4000

Auto-saved library path              : /slowfs/ndmdisk003/r4000_19142
Auto-saved library timestamp         : Wed Aug 14 01:56:21 2019
Auto-saved library size              : 2933585 bytes
Auto-saved library is_recovery_save  : false
Auto-saved library context           : creste_net

prompt> recover_auto_save
Information: Loading library file '/slowfs/ndmdisk003/r4000_19142' (FILE-007)
```

SEE ALSO

stop_auto_save(2)
start_auto_save(2)
remove_auto_save(2)
report_auto_save(2)

recover_rp_placement

Recover placement information for placed relative placement groups.

SYNTAX

int **recover_rp_placement**
rp_group_list | -all
-blockage

Data Types

rp_group_list list or collection

ARGUMENTS

rp_group_list

Specifies a list of relative placement groups. The list may contain relative placement group names, patterns, or collections. A collection may be specified using the **get_rp_groups** command. This option is mutually exclusive with *-all*.

-all

Recovers placement information for all relative placement groups from the current block. This option is mutually exclusive with *rp_group_list*.

-blockage

Specifies that consider relative placement blockages too. Please note that the bounding box of the placement blockages is not recovered. The size of the relative placement blockage is used in recovering placement information of the relative placement groups.

DESCRIPTION

This command recovers placement information for the specified relative placement groups. The placement information includes the bounding box and placement status of the relative placement groups. Note that placement information cannot be recovered if the relative placement group is not placed or placement failed.

EXAMPLES

The following example recovers placement information for relative placement group named "top_rp1" and "top_rp2".

```
prompt> recover_rp_placement { top_rp1 top_rp2 }
2
```

SEE ALSO

create_rp_group(2)
get_rp_groups(2)
add_to_rp_group(2)
remove_from_rp_group(2)
write_rp_groups(2)

recycle_programmable_spare_cells

Recycles the removed ECO cells and converts them back to programmable spare cells (PSC). These PSC cells can be used to do PSC mapping afterward. This command is typically used in the freeze silicon ECO flow.

SYNTAX

status **recycle_programmable_spare_cells**
[-cells *cell_objects*]
[-lib_cells_for_filler_recovery *lib_cells*]

Data Types

cell_objects collection
lib_cells collection

ARGUMENTS

-cells *cell_objects*

Specifies that the spare cells come from removed ECO cells for recycling. The specified cells must have the psc_filler_is_used attribute set to true. If this option is not specified, the command recycles cells by collecting all available spare cells with the psc_filler_is_used attribute set to true from the set of removed ECO cells in the current design.

-lib_cells_for_filler_recovery *lib_cells*

Specifies the programmable spare cell (PSC) filler library cells, The command uses only these library cells to fill and recover the unused space of the PSC cells after PSC recycling. If the option is disabled, the command uses all available PSC filler library cells in the reference libraries.

DESCRIPTION

The **recycle_programmable_spare_cells** command recycles the removed ECO cells and converts them back to PSC cells. These PSC cells can be used to do PSC mapping afterward.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example recycles the cells specified by the **-cells** option and converts them back to PSC cells.

```
prompt> recycle_programmable_spare_cells \
    -cells [get_cells *_Spare]
```

SEE ALSO

place_freeze_silicon(2)

redirect

Redirects the output of a command to a file.

SYNTAX

string **redirect**
[-append] [-tee] [-file | -variable | -channel] [-compress]
[-bg]
[-max_cores *number_of_cores*]
target
{*command_string*}

Data Types

number_of_cores integer
target string
command_string string

ARGUMENTS

-append

Appends the output to *target*.

-tee

Like the Linux command of the same name, sends output to the current output channel as well as to the *target*.

-file

Indicates that *target* is a file name, and redirection is to that file. This is the default.

This option is mutually exclusive with the **-variable** and **-channel** options.

-variable

Indicates that *target* is a variable name, and redirection is to that Tcl variable.

This option is mutually exclusive with the **-file** and **-channel** options.

-channel

Indicates that *target* is a Tcl channel, and redirection is to that channel.

This option is mutually exclusive with the **-file** and **-variable** options.

-compress

Compress when writing to file.

If redirecting to a file, this option specifies that the output should be compressed as it is written. The output file is in a format recognizable by the Linux **gzip -d** command.

You cannot specify this option with the **-append** option.

-bg

Indicates that the redirected command or script executes in the background and in parallel with other foreground commands downstream.

When you run this command, the tool returns the process ID (PID) of the background process and then displays the shell prompt immediately to execute the next command.

The **redirect -bg** command supports only a limited set of commands; the tool issues an error message if you specify an unsupported command. To list the supported commands, use the **list_commands -bg** command. If the command or script includes a **redirect -bg** command, the **-bg** option is ignored.

You can run at most two commands in the background. If there are already two **redirect -bg** commands running, additional **redirect -bg** commands are queued.

If you run the **exit** or **quit** command while background commands are running, the tool waits for all background commands to complete before exiting.

The **-bg** option must be used with the **-file** and **-max_cores** options. No other options are allowed.

-max_cores ***number_of_cores***

Specifies the maximum number of CPU cores that can be used by multicore commands run within the **redirect -bg** command.

target

Indicates the target of the output redirection. This is either a file name, Tcl variable, or Tcl channel depending on the command arguments.

command_string

Specifies the command to execute. Intermediate output from this command, as well as the result of the command, are redirected to *target*. The *command_string* must be rigidly quoted with curly braces.

DESCRIPTION

The **redirect** command performs the same function as the traditional Linux-style redirection operators > and >>. However, it is more flexible because you can redirect multiple commands or an entire script.

By default, the redirected commands are run in the foreground. To run the redirected commands in the background, use the **-bg** option. In both cases, the *command_string* must be rigidly quoted (that is, enclosed in curly braces) for the operation to succeed.

Note that the builtin Tcl command **puts** does not respond to output redirection of any kind. Use the builtin **echo** command instead.

Behavior Without the -bg Option

When you use the **redirect** command without the **-bg** option,

- The **redirect** command returns an empty string.

Screen output occurs only if errors occurred during execution of the *command_string* (other than opening the redirect file). When errors occur, a summary message is printed. See the examples.

Although the result of a successful **redirect** command is the empty string, you can still get and use the result of the redirected command. Construct a **set** command in which you set a variable to the result of your command. Then, redirect the **set** command. The variable holds the result of your command. See the examples.

- By default, output is redirected to a file.

  You can redirect the output to a Tcl variable by using the **-variable** option or to a Tcl channel by using the **-channel** option.

  You can send the output to the current output device as well as the redirect target by using the **-tee** option. See the EXAMPLES section for an example.

Behavior With the -bg Option

The **-bg** option enables parallel processing on multicore hardware for performance reasons. This command is targeted for post-update reporting, analysis, and design export commands.

When you use the **redirect** command with the **-bg** option,

- The command returns the PID of the background job instead of an empty string.
- You must redirect the output to a file; the **-bg** option cannot be used with other redirect targets.
- The maximum number of cores for the main session is reduced by the maximum number of cores used by the background jobs.
- You can use the **parallel_execute** command with the **redirect -bg** command to run jobs in parallel in the background.
- Changes made to the netlist or other data by commands executed in the background job are not brought back to the foreground job.

  Also, using commands in the background job that modify the netlist or other data can cause memory blow up in the child process.
- The background job cannot pass variables back to the foreground job.

  To pass variables from a background job to the foreground job, you must write the contents of the variables to a file in the background job, and then read that file in the foreground job.
- If you use the **change_names** command in the background, you must stop recording Formality setup information before running the **redirect -bg** command and use a new SVF file within the background job. Both the SVF files must be provided for formal verification.

EXAMPLES

In the following example, the output of the plus procedure is redirected. The echoed string and the result of the plus operation are in the output file. Notice that the result is not echoed to the screen.

```
prompt> proc plus {a b} {echo "In plus" ; return [expr $a + $b]}
prompt> redirect p.out {plus 12 13}
prompt> exec cat p.out
In plus
25
```

In this example, a typo in the command creates an error condition. The error message indicates that you can use **error_info** to trace

the error, but you should first check the output file.

```
prompt> redirect p.out {plus2 12 13}
Error: Errors detected during redirect
       Use error_info for more info. (CMD-013)
prompt> exec cat p.out
Error: unknown command 'plus2' (CMD-005)
```

This example shows how to use of the results from redirected commands. Because the result of the **redirect** command for a command that does not generate a Tcl error is the empty string, use the **set** command to trap the result of the command. For example, assume that there is a command to read a file that has a result of "1" if it succeeds, and "0" if it fails. If you redirect only the command, there is no way to know if it succeeded.

```
prompt> redirect p.out { read_a_file "a.txt" }
# Now what? How can I redirect and use the result?
```

If you set a variable to the result, you can use that result in the foreground process.

```
prompt> redirect p.out { set rres [read_a_file "a.txt"] }
if { $rres == 1 } {
  echo "Read ok!"
}
```

The **redirect** command is not limited to redirection of a single command. You can redirect entire blocks of a script with a single **redirect** command. This simple example with the **echo** command demonstrates this feature:

```
prompt> redirect p.out {
?       echo -n "Hello "
?       echo "world"
?     }
prompt> exec cat p.out
Hello world
prompt>
```

The **redirect** command allows you to tee output to the previous output device and also to redirect output to a variable. This simple example with the **echo** command demonstrates this feature:

```
prompt> set y "This is "
This is
prompt> redirect -tee x.out {
        echo XXX
        redirect -variable y -append {
          echo YYY
          redirect -tee -variable z {
            echo ZZZ
          }
        }
      }
XXX
prompt> exec cat x.out
XXX
prompt> echo $y
This is YYY
ZZZ

prompt> echo $z
ZZZ
```

The following example demonstrates how to run a Tcl procedure and a Tcl script in the background by using the **redirect** command

with the **-bg** option. The Tcl procedure and Tcl script can contain any of the commands supported by the **-bg** option.

```
prompt> set_host_options -max_cores 4
prompt> redirect -bg -max_cores 1 -file x.out {MyTclProc}
prompt> redirect -bg -max_cores 1 -file y.out {source y.tcl}
prompt> echo $z
ZZZ
```

The following example demonstrates how the **set_svf** command can be used within the **redirect -bg** command. If the reference design is from foo.v, to verify the netlist after the **redirect -bg** command, use both the compile.svf and redirect.svf SVF files. To verify the netlist after the **compile -incremental** command, use the compile.svf and compile_incr.svf SVF files.

```
prompt> set_host_options -max_cores 4
prompt> set_svf compile.svf
prompt> read_verilog foo.v
prompt> read_sdc foo.sdc
prompt> compile
prompt> set_svf -off
prompt> redirect -bg -max_cores 1 -file x.out \
  {set_svf redirect.svf; change_names -rules verilog -hierarchy; ...}
prompt> set_svf compile_incr.svf
prompt> compile -incremental
ZZZ
```

SEE ALSO

echo(2)
error_info(2)
set(2)
report_background_jobs(2)
parallel_execute(2)

redo

Redo the effects of one or more commands.

SYNTAX

int **redo**
[-levels *num_levels* | -marker *marker_name* | -all]
[-check_only]
[-silent]

Data Types

num_levels int
marker_name string

ARGUMENTS

-levels *num_levels*

Redoes the effects of the specified number of command levels.

-marker *marker_name*

Restores the system state to the point of creation of the marker with the specified name.

-all

Restores the system to the most recent available state in the command history.

-check_only

Causes the command to return the same status and messages it would otherwise return without actually performing the redo or changing anything.

-silent

Surpresses messages and a TCL error in the event of an error.

DESCRIPTION

This command redoes the effects of one or more previous database-changing commands, or "command levels". If the *-levels* option is used, then the specified number of command levels will be redone. It is an error to specify more levels than there are in the command history that are more recent than the current point.

If the *-marker* option is used, then the system state will be restored to that of when the specified marker was created. *marker_name* may be the user assigned or system generated name of the marker. This option would typically be used to restore the system to a user marker, which would have been created using the *create_undo_marker* command. It is an error to specify a marker that was created previous to the current point in the command history - for that case use the *undo* command instead.

If the *-all* option is used, then the system state will be restored to the latest available state in the command history. Note that this may not be the latest state that was originally undone from, as the execution of a new state-changing command earlier in the history will cause the subsequent redo history to be truncated.

If neither the *-levels*, *-marker*, or *-all* option is used, then this command will redo one command level.

The command returns the actual number of command levels redone or, if the *-check_only* option was used, that would have been redone without the *-check_only* option.

EXAMPLES

The following redoes one command level:

```
prompt> get_attribute $inst origin
0.0000 0.0000
prompt> set_attribute $inst origin {100 300}
{U368}
prompt> get_attribute $inst origin
100.0000 300.0000
prompt> undo
1
prompt> get_attribute $inst origin
0.0000 0.0000
prompt> redo
1
prompt> get_attribute $inst origin
100.0000 300.0000
```

The following returns the system to the most recent state:

```
prompt> create_undo_marker my_mrk
my_mrk
prompt> get_attribute $inst origin
0.0000 0.0000
prompt> set_attribute $inst origin {200 400}
{U368}
prompt> set_attribute $inst origin {300 600}
{U368}
prompt> get_attribute $inst origin
300.0000 600.0000
prompt> undo -marker my_mrk
2
prompt> get_attribute $inst origin
0.0000 0.0000
prompt> redo -all
```

```
2
prompt> get_attribute $inst origin
300.0000 600.0000
```

SEE ALSO

undo(2)
create_undo_marker(2)
eval_with_undo(2)
get_undo_info(2)

refine_placement

Performs incremental placement with congestion optimization.

SYNTAX

status **refine_placement**
[-effort very_low | medium | high]
[-congestion_effort low | medium | high]
[-perturbation_level low | medium | high]
[-coordinates *bbox*]

Data Types

bbox rectangle

ARGUMENTS

-effort very_low | low | medium | high

Specifies the CPU effort level for coarse placement. If running very low effort it is advised to set the -congestion_effort to low. The default effort level is medium.

-congestion_effort low | medium | high

Specifies the effort level of congestion removal. The higher effort, the less attention is paid to other objects: timing, density and total wire-length. The default effort level is **medium**.

-perturbation_level low | medium | high

Specifies the perturbation level for the placement to optimize congestion. Specifying **low** produces the least amount of movement from the placement. The higher the level specified, the more movement you might expect from the placement. So, specifying **high** produces the most movement from the placement. The default effort level is **medium**.

-coordinates *bbox*

Specifies one rectangle that defines the region from which to run incremental congestion removal.

The "bbox" is lower-left and upper-right coordinates of a rectangle.

The default (without this option) is to run incremental placement for the entire design, which is the recommended way to use this command. If you want to specify one localized region on which incremental placement should be done, you can use this option to prevent any disturbance of the rest of the converged design. The region based refinement may cause timing/congestion degradation if regions are not properly specified.

The units of the **-coordinates** are in microns.

No runtime improvement is expected when this option is used; it is primarily used to restrict the placer from making changes outside the specified rectangles.

DESCRIPTION

The **refine_placement** command performs incremental congestion optimization for the current design. This command does not touch netlist.

Multicorner-Multimode Support

This command operates a scenario selected based on timing QoR.

EXAMPLES

The following example runs the **refine_placement** command in low-effort congestion mode:

```
prompt> refine_placement -congestion_effort low
```

The following example runs the **refine_placement** command in single region:

```
prompt> refine_placement -coordinates { {$llx1 $lly1} { $urx1 $ury1 } }
```

SEE ALSO

create_placement(2)

refine_topology_plans

Refine a topology plan to create additional nodes and edges.

SYNTAX

int **refine_topology_plans**
[*topology_plan_list*]

Data Types

topology_plan_list collection

ARGUMENTS

topology_plan_list

Specifies the list of topology plans to refine. If not specified then it means all the topology plans within the current design.

DESCRIPTION

The **refine_topology_plans** command refines a topology plan by creating additional nodes and edges.

If a topology edge has layer-shapes attribute defined with Manhattan style edge segments, then new topology nodes are created where the edge segments cut at a block boundaries. The original topology edge is also cut at the position where the new topology nodes are created, and new topology edges are created between new and existing topology nodes, and then connect them together.

If the topology edge does not have the layer-shapes attribute defined, or the layer-shapes are not Manhattan style, then there will be no new nodes or edges created for this topology edge.

Upon success, the command returns 1. Otherwise, 0.

EXAMPLES

The following example refines all the topology plans of the design.

```
prompt> refine_topology_plans
```

SEE ALSO

create_topology_plans(2)

refresh_performance_via_ladder_constraints

Refreshes via ladder constraints in a design to ensure compliance with via ladder candidates.

SYNTAX

status **refresh_performance_via_ladder_constraints**

DESCRIPTION

The **refresh_performance_via_ladder_constraints** command refreshes via ladder constraints in a design to ensure compliance with via ladder candidates.

EXAMPLES

The following example verifies all via ladder constraints in the design and updates via ladder constraints which comply with via ladder candidates.

```
prompt> refresh_performance_via_ladder_constraints
```

SEE ALSO

insert_via_ladders(2)
refresh_via_ladders(2)
remove_via_ladder_constraints(2)
remove_via_ladders(2)
set_via_ladder_constraints(2)
verify_via_ladders(2)

refresh_via_ladders

Refreshes via ladders to ensure compliance with via ladder constraints. This command runs the **verify_via_ladders** command, removes any via ladders that would cause "Misplaced via ladder" DRCs, and runs the **insert_via_ladders** command to insert via ladders where needed. The command is further controlled with the route.auto_via_ladder.* application options.

SYNTAX

status **refresh_via_ladders**
[-nets {*collection_of_nets*}]

Data Types

collection_of_nets collection

ARGUMENTS

-nets {*collection_of_nets*}

Specifies the nets on which to update via ladders. A via ladder will be updated only on pins with via ladder constraints when the pin is also on one of the specified nets.

If you do not specify this option, via ladders are updated on any pin with a via ladder constraint.

DESCRIPTION

The **refresh_via_ladders** command refreshes via ladders in a design to ensure compliance with via ladder constraints. This command runs the **verify_via_ladders** command, removes any via ladders that would cause "Misplaced via ladder" DRCs, and runs the **insert_via_ladders** command to insert via ladders where needed.

The **route.auto_via_ladder.update_during_route** application option, when true, activates an automatic via ladder update at the start of all routing commands that run global routing, track assignment, and detail routing. This includes the **route_auto**, **route_group**, and **route_eco** commands.

The **refresh_via_ladders** command executes the same internal flow as automatic via ladder updating in routing commands, but does so as a standalone command. The **refresh_via_ladders** command performs the via ladder update and terminates immediately without performing any other routing tasks.

EXAMPLES

The following example verifies all via ladders in the design, removes any via ladders that cause "Misplaced via ladder DRCs," and performs via ladder insertion. Vias ladders are inserted on any pin with a via ladder constraint but no corresponding via ladder in the layout.

```
prompt> refresh_via_ladders
```

SEE ALSO

insert_via_ladders(2)
remove_via_ladder_constraints(2)
remove_via_ladders(2)
set_via_ladder_constraints(2)
set_via_ladder_rules(2)
verify_via_ladders(2)

remove_3d_virtual_blocks

Removes virtual interface blocks.

SYNTAX

integer **remove_3d_virtual_blocks**
-all | *vib* | *vib_cell* | {*die1 die2*}

Data Types

vib collection
vib_cell collection
die1 collection
die2 collection

ARGUMENTS

-all

Removes all virtual interface blocks in the design.

vib

Specifies the virtual interface block to remove.

vib_cell

Specifies the virtual interface block by cell instance to remove.

{ *die1 die2* }

Specifies the dies between which to remove the virtual interface block.

DESCRIPTION

This command removes virtual interface blocks from the design. If the virtual interface block was connected to the top netlist during creation, the die ports at the interface are reconnected back together.

EXAMPLES

The following example removes all virtual interface blocks from the design.

```
prompt> remove_3d_virtual_blocks -all
```

The following example removes the virtual interface block between two dies.

```
prompt> remove_3d_virtual_blocks {die1 die2}
```

The following example removes the specified virtual interface block.

```
prompt> remove_3d_virtual_blocks die1__die2
```

SEE ALSO

create_3d_virtual_blocks(2)

remove_abstract

Removes the abstract associated with the given design from memory and disk

SYNTAX

return_val **remove_abstract**
[*design_name*]

Data Types

return_val 0 (or 1) if command failed (or succeeded).
design_name Name in form [libName:]designName/label

ARGUMENTS

design_name

The source design name with optional library specifications. If the option is not given, the **current_design** is used. If the library specification is not given, the **current_lib** is used. The library of the specified block should already be opened in edit mode.

RETURN VALUE

Returns 1 if the design is saved and 0 if not. If an illegal name(with a slash) is given for the design, a TCL error is raised.

DESCRIPTION

The command removes the abstract view associated with the given (or current) design. If the abstract is editable, the open blocks on which the abstract depends will automatically be upgraded to "edit" mode.

EXAMPLES

Remove the abstract view associated with the current design.

```
prompt> remove_abstract
1
```

Following example removes the abstract associated with the design "mid", in library "midlib".

```
prompt> list_blocks
Lib toplib
  +>  0 top.design current
Lib midlib
  ->  0 mid.abstract
  -   0 mid.design
3
prompt> remove_abstract midlib:mid
1
prompt> list_blocks
Lib toplib
  +>  0 top.design current
Lib midlib
  ->   0 mid.design
2
```

SEE ALSO

create_abstract(2)
merge_abstract(2)

remove_annotated_check

Removes annotated timing-check information.

SYNTAX

status **remove_annotated_check**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
-all | -from *from_list* | -to *to_list*
[-rise | -fall]
[-clock rise | fall]
[-setup]
[-hold]
[-recovery]
[-removal]
[-nochange_low]
[-nochange_high]
[-width]
[-period]

Data Types

from_list list
to_list list

ARGUMENTS

-modes *mode_list*

Specifies the modes for which to remove the annotated value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is de-annotated separately. If this option is not given, the command removes annotation for every scenario that is active with the specified corner(s). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to remove the annotated value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is de-annotated separately. If this option is not given, the command removes annotation for every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios for which to remove the annotated value. Each of the specified scenarios is de-annotated separately. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command applies to the current scenario only.

-all

Removes all annotated checks in the current design. This includes rise and fall values for setup, hold, recovery, removal, and nochange pin-to-pin timing checks. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments or options.

-from ***from_list***

Specifies a list of leaf cell clock pins that are the startpoints of the timing arcs from which annotated checks are to be removed. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments or options.

-to ***to_list***

Specifies a list of leaf cell data pins that are the endpoints of the timing arcs from which annotated checks are to be removed. You must specify either **-all**, **-from**, or **-to**; if you specify **-all**, you cannot specify any other arguments or options.

-rise | **-fall**

Specifies whether the check to be removed is for data rise or fall transition. If you do not specify either **-rise** or **-fall**, both values are removed.

-clock rise | **fall**

Specifies whether the check to remove is for clock rising or falling. By default, checks for both clock rise and fall are removed.

-setup

Removes only setup timing-check information.

-hold

Removes only hold timing-check information.

-recovery

Removes only recovery timing-check information.

-removal

Removes only removal timing-check information.

-nochange_low

Removes only the nochange timing check against data low information.

-nochange_high

Removes only the nochange timing check against data high information.

-width

Removes only minimum pulsewidth timing check information.

-period

Removes only period timing check information.

DESCRIPTION

This command removes annotated timing checks between the specified pins. Data rise and fall and clock rise and fall annotated checks are removed by default.

You can use the **remove_annotated_check** command for pins at lower levels of the design hierarchy. These pins are specified as "INSTANCE1/INSTANCE2/PIN_NAME."

If the design is not already linked, **remove_annotated_check** links it automatically.

The **remove_annotated_check** command removes annotated timing checks set by the **set_annotated_check** command, and also removes setup, hold, recovery, removal, or nochange annotated timing checks set by the **read_sdf** command. The **reset_design** command removes annotated timing checks in addition to other information.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes an annotated setup check between pins u1/u2/CP and u1/u2/D:

```
prompt> remove_annotated_check -setup -from u1/u2/CP -to u1/u2/D
```

The following example removes all annotated timing checks on the current design:

```
prompt> remove_annotated_check -all
```

SEE ALSO

current_design(2)
link(2)
read_sdf(2)
report_annotated_check(2)
reset_design(2)
set_annotated_check(2)

remove_annotated_delay

Removes the annotated delay between two pins.

SYNTAX

status **remove_annotated_delay**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
-all | -from *from_list* | -to *to_list*

Data Types

from_list list
to_list list

ARGUMENTS

-modes *mode_list*

Specifies the modes for which to remove the annotated value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is de-annotated separately. If this option is not given, the command removes annotation for every scenario that is active with the specified corner(s). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to remove the annotated value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is de-annotated separately. If this option is not given, the command removes annotation for every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios for which to remove the annotated value. Each of the specified scenarios is de-annotated separately. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command applies to the current scenario only.

-all

Removes all net- and cell-annotated delays in the current design. If you specify **-all**, you cannot specify any other arguments.

-from *from_list*

Specifies a list of leaf-cell pins or top-level ports that are the startpoints of the timing arcs from which to remove annotated delays. You must specify either **-all**, or both **-from** and **-to**.

-to *to_list*

Specifies a list of leaf-cell pins or top-level ports that are the endpoints of the timing arcs from which to remove annotated delays. You must specify either **-all**, or both **-from** and **-to**.

DESCRIPTION

This command removes annotated delays between the specified pins, which must be on the same net or on the same cell. Both rise and fall annotated delays are removed. There must be a timing arc between the pins in *from_list* and the pins in *to_list* or no annotated delay is removed.

You can use **remove_annotated_delay** for pins at lower levels of the design hierarchy. These pins are specified as "INSTANCE1/INSTANCE2/PIN_NAME."

To annotate net delay values between pins in a design, use **set_annotated_delay -net** command. To annotate cell delay values between pins in a design, use the **set_annotated_delay -cell** command. You must use **-from** and **-to** in the same manner you used them to set annotated values.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example first shows the annotation of the delay to remove. The second command removes the annotated net delay between output pin Z of cell instance U1/U2/U3 and input pin A of cell instance U6:

```
prompt> set_annotated_delay -net <value> -from U1/U2/U3/Z -to U6/A
prompt> remove_annotated_delay -from U1/U2/U3/Z -to U6/A
```

The following example removes all annotated net and cell delays from the current design:

```
prompt> remove_annotated_delay -all
```

SEE ALSO

current_design(2)
report_timing(2)
reset_design(2)
set_annotated_delay(2)

remove_annotated_power

Removes annotated cell power dissipation values that were set by **set_annotated_power**.

SYNTAX

status **remove_annotated_power**
-scenarios *scenario_list*
-all
[cell_list]

Data Types

scenario_list list
cell_list list

ARGUMENTS

-scenarios

Specifies the set of scenarios for which the annotated values should be removed. The default is the value returned by the **current_scenario** command.

-all

Specifies that the power annotations of all cells should be removed.

cell_list

Specifies the cells for which the annotated power values should be removed. When option *-all* is given, no cell_list is allowed.

DESCRIPTION

This command removes annotated power values on cells. These annotations were previously applied using the *set_annotated_power* command. The command removes all annotations of a cell, so both leakage and internal power annotations for all supply nets are removed.

Multicorner-Multimode Support

EXAMPLES

The following example removes the power annotations for cell U1/U2/U3 in the current scenario.

```
prompt> remove_annotated_power U1/U2/U3
```

The following example removes the power annotations of all cells in the scen1 scenario.

```
prompt> remove_annotated_power -all -scenarios scen1 \
```

SEE ALSO

report_annotated_power(2)
set_annotated_power(2)

remove_annotated_transition

Removes the annotated transition at a pin.

SYNTAX

status **remove_annotated_transition**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
-all | *pins*

Data Types

pins collection

ARGUMENTS

-modes *mode_list*

Specifies the modes for which to remove the annotated value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is de-annotated separately. If this option is not given, the command removes annotation for every scenario that is active with the specified corner(s). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to remove the annotated value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is de-annotated separately. If this option is not given, the command removes annotation for every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios for which to remove the annotated value. Each of the specified scenarios is de-annotated separately. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command applies to the current scenario only.

-all

Removes the annotated transition time from every pin in the current design. If you specify **-all**, you cannot specify any other arguments.

pins

Removes the annotated transition time from the specified pins. If you specify **-all**, you cannot specify any other arguments.

DESCRIPTION

This command removes annotated transitions at the specified pins. Both rise and fall annotated transitions are removed.

To annotate net-transition values at pins in a design, use the **set_annotated_transition** command.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes the annotated transition at every pin in the design:

```
prompt> remove_annotated_transition -all
```

The following example first shows the annotation of the transition to be removed. The second command removes the annotated transition at the output pin Z of cell instance U1/U2/U3:

```
prompt> set_annotated_transition value U1/U2/U3/Z
prompt> remove_annotated_transition U1/U2/U3/Z
```

SEE ALSO

current_design(2)
report_timing(2)
reset_design(2)
set_annotated_transition(2)

remove_antenna_rules

Name

remove_antenna_rules

Removes the antenna rules in the library.

SYNTAX

integer **remove_antenna_rules**
[-mode *antenna_mode*]
[-name *rule_name*]
[-library *library*]

Data Types

antenna_mode integer
rule_name string
library collection

ARGUMENTS

-mode *antenna_mode*

An integer value between 1 and 6, that represents the way the antenna areas are computed. When this option is specified, only the antenna rules that use the given mode for antenna area calculation, are updated to remove the specified mode. This is an optional option and mutually exclusive to *-name*.

-name *rule_name*

The name of the antenna rule to be deleted from the given library. The command aborts if an antenna rule with the given name does not exist. This is an optional option and mutually exclusive to *-mode*.

-library *library*

The library from which the given antenna rules are to be deleted. This is an optional option. When not specified, the antenna-rules from the current library are deleted.

DESCRIPTION

This command removes the antenna rules from the given library. The default behavior is to remove all the antenna rules from the current library.

The command returns the number of antenna rules removed from the given library when all rules are removed. It returns 1 when a rule is removed based on a specified name or when a mode is removed from corresponding rules.

EXAMPLES

The following example removes the antenna rule 'r1' from the current library.

```
prompt>remove_antenna_rules -name r1
1
```

The following example removes all the antenna rules from the library 'lib1'

```
prompt>remove_antenna_rules -library lib1
16
```

SEE ALSO

define_antenna_rule(2)
define_antenna_layer_rule(2)
get_antenna_rule_names(2)
report_antenna_rules(2)

remove_array_from_macro_group

Remove macro array to macro group.

SYNTAX

status **remove_array_from_macro_group**
-array macro_array

ARGUMENTS

-macro_array macro_array

Specifies macro array.

DESCRIPTION

This command remove specified macro array from the macro group it belongs to. All macros of the macro array will be removed from the macro group.

EXAMPLES

The following command remove macro array array1 from its macro group.

prompt> **remove_array_from_macro_group** -array array1

SEE ALSO

create_macro_group(2)
add_array_from_macro_group(2)
add_macro_to_group(2)
remove_macros_from_group(2)
clone_macro_group_packing(2)
create_macro_array(2)

remove_attachments

Removes the given attachments from the specified owner object.

SYNTAX

status **remove_attachments**
names
[-of_object *objects*]

Data Types

names list
objects collection

ARGUMENTS

names

List of attachment names to be removed. This argument supports glob style patterns.

-of_object *objects*

Specifies the object whose attachments will be removed. By default, the current block and current library will be considered in that order as owner object.

DESCRIPTION

This command removes the attachments with the names in the given list from the objects provided through the **-of_object** option. If **-of_object** is not used, then the current block is used as the owner object. If current block is not set, current library is used as the owner object For the attachment names, glob style pattern is supported.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the attachment test_attachment from the current block.

```
prompt> remove_attachments -of_object [current_block] test_attachment
1
```

SEE ALSO

add_attachment(2)
open_attachment(2)
report_attachments(2)

remove_attributes

Removes an attribute from the specified objects.

SYNTAX

collection **remove_attributes**
[-quiet]
-objects *object_list*
-name *attribute_name*
pos_object_list
pos_attribute_name

object_list list
attribute_name string
pos_object_list list
pos_attribute_name string

ARGUMENTS

-quiet

Turns off the warning messages that would otherwise be issued if the attribute or objects are not found.

-objects *object_list*

Specifies a list of objects from which the attribute is to be removed. Each element in the list is a collection of objects. This option is mutually exclusive with *pos_object_list*. Either one (and only one) of *-objects* or *pos_object_list* must be specified.

-name *attribute_name*

Specifies the name of the attribute to be removed. This option is mutually exclusive with *pos_attribute_name*. Either one (and only one) of *-name* or *pos_attribute_name* must be specified.

pos_object_list

Specifies a list of objects from which the attribute is to be removed. Each element in the list is a collection of objects. This positional option is provided for compatibility with other tools.

pos_attribute_name

Specifies the name of the attribute to be removed. This positional option is provided for compatibility with other tools.

DESCRIPTION

This command removes the specified attribute from the specified objects. This command returns a collection of objects that have the specified attribute removed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the attributes. However, for attributes such as **max_leakage_power**, which are scenario dependent, this command uses information from the current scenario only.

EXAMPLES

The following command defines a new attribute named *X* on the net:

```
prompt> define_user_attribute -type int -classes net -name X
1
```

The following command sets the *X* attribute to 30 on all the nets in the current hierarchy:

```
prompt> set_attribute -objects [get_nets *] -name X -value 30
{"out", "in", "VDD", "VSS"}
```

This command removes the *X* attribute from the nets named *VSS* and *VDD*:

```
prompt> remove_attributes -objects [get_nets V*] -name X
{"VDD", "VSS"}
```

This command retrieves the *X* attribute from the net named *VDD*:

```
prompt> get_attribute -objects [get_nets VDD] -name X
Warning: Attribute 'X' does not exist on net 'VDD' (ATTR-3)
```

The following command retrieves the *X* attribute from a net named *in*;

```
prompt> get_attribute -objects [get_nets in] -name X
30
```

This command removes the *output_not_used* attribute on a port:

```
prompt> remove_attributes -objects [get_ports OUT1] -name output_not_used
{"OUT1"}
```

In the following example, a warning message is issued because the cell named *U9* cannot be found:

```
prompt> remove_attributes -objects [get_cells {U1 U9}] -name dont_touch
Warning: No cell objects matched 'U9' (SEL-004)
{"U1"}
```

In the following example, the specified attribute cannot be found:

```
prompt> remove_attributes -objects [get_cells *] -name arrival
Warning: Attribute 'arrival' has not been defined for cells (ATTR-1)
```

SEE ALSO

define_user_attribute(2)
get_attribute(2)
get_defined_attributes(2)
list_attributes(2)
report_attributes(2)
set_attribute(2)
shell.dc_compatibility.remove_libcell_attribute(3)

remove_auto_save

Removes the auto-saved library of the specified library.

SYNTAX

status **remove_auto_save**
[-library library_name | library_path]

Data Types

library_name string
library_path string
auto_saved_library_name string

ARGUMENTS

-library

Specifies the library name or path whose auto-saved library needs to be removed.

DESCRIPTION

This command removes the specified auto-saved library of the specified library. By default, this command removes the auto-saved library of the current library.

EXAMPLES

The following example removes the auto-saved library of the current library.

prompt> **remove_auto_save**

SEE ALSO

stop_auto_save(2)

recover_auto_save(2)
start_auto_save(2)
report_auto_save(2)

remove_blackbox_timing

Removes black box timing data and updates the abstract view for the current block.

SYNTAX

status **remove_blackbox_timing**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command removes the black box timing information on the current block. The abstract view of the current block is updated or re-created to reflect the change.

EXAMPLES

The following example creates an abstract view with black box timing information for the current design, and then updates the abstract view to remove BBT data.

```
prompt> create_clock -period 10 -name main_clock [get_ports Clk]
prompt> set_blackbox_clock_port Clk
prompt> create_blackbox_load_type -lib_cell buf0 loadType1
prompt> create_blackbox_drive_type -lib_cell buf8 driveType1
prompt> set_blackbox_port_load -type loadType1 [all_inputs]
prompt> set_blackbox_port_drive -type driveType1 [all_outputs]
prompt> create_blackbox_delay -rise_from Clk -to [all_outputs] -clock main_clock -value 0.3
prompt> create_blackbox_constraint -from Clk -edge rise -to [all_inputs] \
    -clock main_clock -value 0.3
prompt> commit_blackbox_timing
prompt> remove_blackbox_timing
```

SEE ALSO

commit_blackbox_timing(2)

remove_block_pin_constraints

Removes the existing block pin constraints.

SYNTAX

string **remove_block_pin_constraints**
[-cells *cell_collection*]
[-self]
[-pin_spacing_control]

Data Types

cell_collection collection

ARGUMENTS

-cells *cell_collection*

Specifies the blocks from which to remove block pin constraints. If you specify this option, the command only removes the block pin constraints for the specified blocks.

If you do not specify this option and "-self" option, the command removes block pin constraints for all blocks inside current design.

-self

If this option is specified, the command removes the block pin constraints that are associated with current top design.

-pin_spacing_control

If this option is specified, the command removes the block pin spacing constraints that are set by read_pin_constraints and associated with reference blocks.

DESCRIPTION

This command removes block pin constraints from blocks. If you do not specify any arguments, this command removes pin constraints for all blocks.

The **set_editability** command can enable or disable the **remove_block_pin_constraints** command for specified blocks or levels of physical hierarchy. This command has no effect on the blocks that are disabled with the **set_editability** command.

EXAMPLES

The following example removes the physical constraints for all pins, nets and blocks.

```
prompt> remove_block_pin_constraints
```

SEE ALSO

report_block_pin_constraints(2)
set_block_pin_constraints(2)
set_editability(2)

remove_blocks

Removes one or more open blocks from memory, its library, and the disk.

SYNTAX

return_val **remove_blocks**
[-force]
blocks

Data Types

return_val 0 (or 1) if command failed (or succeeded)
blocks A single block name in [libName:]blockName[/labelName][.viewName] format or a collection of blocks

ARGUMENTS

-force

This option is used when the destination is (or may be) open and modified but not yet saved.

blocks

A block name with optional library, label, and view specifications, or a collection of blocks. If the library specification is not present, the **current_lib** is used. If the label specification is not present, the default un-named label is used. If the view specification is not present, then *design view* is used. If the option is not given, the **current_block** is used. When no view is explicitly specified then all available views are considered.

RETURN VALUE

Returns the number of blocks removed and 0 if not. If an illegal name(with a slash) is given for the block, a Tcl error is raised.

DESCRIPTION

This command removes an open block and all of its contents. After it is executed the block is purged from memory and its disk contents are removed. USE THIS COMMAND WITH EXTREME CAUTION! It is intended for use primarily in scripts that (re)construct blocks at the beginning to reinitialize a block completely from scratch. The command will fail if the block has been modified but not yet saved unless the **-force** option is given. Note that this command works if a block exists whether it is in memory or not.

When no view is explicitly specified for **blocks** then all available views of source block are removed.

EXAMPLES

This example forcibly removes the block Top from both memory and from disk.

```
prompt> remove_blocks -force Top
1
```

SEE ALSO

close_blocks(2)
copy_block(2)
current_block(2)
current_design(2)
get_blocks(2)
get_designs(2)
move_block(2)
open_block(2)
open_lib(2)
remove_blocks(2)
reopen_block(2)

remove_bound_shapes

Removes bound shapes from the current design.

SYNTAX

int **remove_bound_shapes**
[-verbose]
[-all]
[-force]
[*bound_shape_list*]

Data Types

bound_shape_list list

ARGUMENTS

-verbose

Displays verbose deletion information.

-all

Removes all bound shapes and their owning move bounds from the current design.

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

bound_shape_list

Specifies a list of bound shapes to remove. The list may contain bound shape collections, specified using the **get_bound_shapes** command.

DESCRIPTION

This command removes bound shapes from the current design except macro group move bound's bound shapes. If all bound shapes of a particular move bound is removed, the move bound is removed as well.

Macro group move bound's shapes cannot be removed using this command. To remove bound shapes from macro group move

bound use remove_bound to completely remove the bound followed by create bound to create the macro_group move bound with desired bound shapes.

The number of removed bound shapes are returned.

EXAMPLES

The following example removes bound shapes of bound "movebound1".

```
prompt> remove_bound_shapes [get_bound_shapes -of_objects movebound1]
2
```

The following example removes all bound shapes in the design.

```
prompt> remove_bound_shapes -all
5
```

SEE ALSO

create_bound_shape(2)
get_bound_shapes(2)
create_bound(2)
get_bounds(2)
add_to_bound(2)
remove_from_bound(2)
report_bounds(2)

remove_boundary_cell_rules

Remove the boundary cell insertion and checking rules.

SYNTAX

status **remove_boundary_cell_rules**
[-all]
[-left_boundary_cell]
[-right_boundary_cell]
[-bottom_boundary_cells]
[-top_boundary_cells]
[-bottom_left_outside_corner_cell]
[-bottom_right_outside_corner_cell]
[-top_left_outside_corner_cell]
[-top_right_outside_corner_cell]
[-bottom_left_inside_corner_cells]
[-bottom_right_inside_corner_cells]
[-top_left_inside_corner_cells]
[-top_right_inside_corner_cells]
[-mirror_left_outside_corner_cell]
[-mirror_right_outside_corner_cell]
[-mirror_left_inside_corner_cell]
[-mirror_right_inside_corner_cell]
[-mirror_left_boundary_cell]
[-mirror_right_boundary_cell]
[-mirror_left_inside_horizontal_abutment_cell]
[-mirror_right_inside_horizontal_abutment_cell]
[-tap_distance]
[-top_tap_cell]
[-bottom_tap_cell]
[-prefix]
[-separator]
[-insert_into_blocks]
[-at_va_boundary]
[-no_1x]
[-min_row_width]
[-min_horizontal_jog]
[-min_vertical_jog]
[-min_horizontal_separation]
[-min_vertical_separation]
[-do_not_swap_top_and_bottom_inside_corner_cell]

ARGUMENTS

-all

Specifies whether to remove all the existed boundary cell rules.

-left_boundary_cell

Specifies whether to remove the existed left boundary cell rule.

-right_boundary_cell

Specifies whether to remove the existed right boundary cell rule.

-bottom_boundary_cells

Specifies whether to remove the existed bottom boundary cells rule.

-top_boundary_cells

Specifies whether to remove the existed top boundary cells rule.

-bottom_left_outside_corner_cell

Specifies whether to remove the existed bottom left outside corner cell rule.

-bottom_right_outside_corner_cell

Specifies whether to remove the existed bottom right outside corner cell rule.

-top_left_outside_corner_cell

Specifies whether to remove the existed top left outside corner cell rule.

-top_right_outside_corner_cell

Specifies whether to remove the existed top right outside corner cell rule.

-bottom_left_inside_corner_cells

Specifies whether to remove the existed bottom left inside corner cells rule.

-bottom_right_inside_corner_cells

Specifies whether to remove the existed bottom right inside corner cells rule.

-top_left_inside_corner_cells

Specifies whether to remove the existed top left inside corner cells rule.

-top_right_inside_corner_cells

Specifies whether to remove the existed top right inside corner cells rule.

-mirror_left_outside_corner_cell

Specifies whether to remove the existed mirror left outside corner cell rule.

-mirror_right_outside_corner_cell

Specifies whether to remove the existed mirror right outside corner cell rule.

-mirror_left_inside_corner_cell

Specifies whether to remove the existed mirror left inside corner cell rule.

-mirror_right_inside_corner_cell

Specifies whether to remove the existed mirror right inside corner cell rule.

-mirror_left_boundary_cell

Specifies whether to remove the existed mirror left boundary cell rule.

-mirror_right_boundary_cell

Specifies whether to remove the existed mirror right boundary cell rule.

-mirror_left_inside_horizontal_abutment_cell

Specifies whether to remove the existed mirror left inside horizontal abutment cell rule.

-mirror_right_inside_horizontal_abutment_cell

Specifies whether to remove the existed mirror right inside horizontal abutment cell rule.

-tap_distance

Specifies whether to remove the existed tap distance rule. If the tap distance rule is to be removed, the top tap cell and bottom tap cell are also removed.

-top_tap_cell

Specifies whether to remove the existed top tap cell rule. If the top tap cell rule is to be removed, the bottom tap cell rule is also to be removed or is not existed in current design, then the tap distance rule is also to be removed.

-bottom_tap_cell

Specifies whether to remove the existed bottom tap cell rule. If the bottom tap cell rule is to be removed, the top tap cell rule is also to be removed or is not existed in current design, then the tap distance rule is also to be removed.

-prefix

Specifies whether to remove the existed prefix rule.

-separator

Specifies whether to remove the user specified separator. If yes, the default value "!" will be used.

-insert_into_blocks

Specifies whether to remove the insert into blocks rule.

-at_va_boundary

Specifies whether to remove the existed at va boundary rule.

-no_1x

Specifies whether to remove the existed no 1x rule. If any of top right outside corner cell rule, bottom right outside corner cell rule, top left outside corner cell rule and bottom left outside corner cell rule is to be removed, then no 1x rule is also to be removed.

-min_row_width

Specifies whether to remove the existed min row width rule.

-min_horizontal_jog

Specifies whether to remove the existed min horizontal jog rule.

-min_vertical_jog

Specifies whether to remove the existed min vertical jog rule.

-min_horizontal_separation

Specifies whether to remove the existed min horizontal separation rule.

-min_vertical_separation

Specifies whether to remove the existed min vertical separation rule.

-do_not_swap_top_and_bottom_inside_corner_cell

Specifies whether to remove the existed do no swap top and bottom inside corner cell rule.

DESCRIPTION

This command is used to remove the existed boundary cell insertion and checking rules. If user specifies one option whose corresponding boundary cell rule is not stored in design, then the option will be automatically ignored.

EXAMPLES

The following example removes all the existed boundary cell rules.

prompt> **remove_boundary_cell_rules -all**

SEE ALSO

set_boundary_cell_rules(2)
report_boundary_cell_rules(2)
compile_boundary_cells(2)
check_boundary_cells(2)
compile_targeted_boundary_cells(2)
check_targeted_boundary_cells(2)

remove_boundary_optimization

Removes the boundary_optimization restriction on specified pins, cells or modules.

SYNTAX

int **remove_boundary_optimization**
object_list

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of pins, cells or modules where to remove boundary optimization restrictions. Cell or reference names in object_list must be from the current design. If more than one object is specified, they must be enclosed in quotation marks ("") or braces ({}).

DESCRIPTION

This command removes the boundary_optimization restriction on objects in object_list, which were previously set using set_boundary_optimization command. The command has no effect on objects specified in the object_list, which were not restrictred earlier.

EXAMPLES

The following example shows how to remove previously set boundary optimization restriction on the cells named U0, U1 and U2.

prompt> remove_boundary_optimization { U0 U1 }

prompt> remove_boundary_optimization U2

SEE ALSO

compile(2)
set_boundary_optimization(2)
report_boundary_optimization(2)

remove_bounds

Removes bounds from the current design.

SYNTAX

int **remove_bounds**
[-force]
[-verbose]
[-all]
[*bound_list*]

Data Types

bound_list list

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Displays verbose deletion information.

-all

Removes all bounds from the current design.

bound_list

Specifies a list of bounds to remove. The list may contain bound names, patterns, or collections. A collection may be specified by using the **get_bounds** command.

DESCRIPTION

This command removes bound constraints from the current design. The number of removed bounds are returned.

EXAMPLES

The following example removes bounds named "movebound1" and "movebound2".

```
prompt> remove_bounds { movebound1 movebound2 }
2
```

The following example removes bounds that match the pattern "movebound*".

```
prompt> remove_bounds movebound*
3
```

The following example removes all bounds in the design.

```
prompt> remove_bounds -all
5
```

SEE ALSO

create_bound(2)
get_bounds(2)
add_to_bound(2)
remove_from_bound(2)
report_bounds(2)

remove_buffer

Removes the buffer cells at a specified driver pin or net on a mapped design.

SYNTAX

status **remove_buffer**
-from *start_point*
-net *net_list*
cell_list
[-allow_tie_connection]

Data Types

start_point	list
net_list	list
cell_list	list
allow_tie_connection	boolean

ARGUMENTS

-from *start_point*

Specifies the starting pin or port of the fanout tree from which the buffers or inverter pairs need to be removed. Only driver pins or driver ports can be specified for this option.

This option is mutually exclusive with the **-net** and **cell_list** options.

-net *net_list*

Specifies the starting net of the fanout tree from which the buffers or inverter pairs need to be removed. Only one net can be specified for this option and the driver pin of this net would be considered as the start for fanout traversal. This means that all of the load cells connected to the net are considered in the buffer tree.

This option is mutually exclusive with the **-from** and **cell_list** options.

cell_list

Specifies the buffer or inverter cells to be removed. Each buffer is a cell name or a collection of netlist cells. When inverter cells are specified, they must be specified in pairs.

This option is mutually exclusive with **-from** options.

-allow_tie_connection

Specifies whether the buffers connected with the tie/PG net can be removed.

DESCRIPTION

This command removes buffer cells when you specify a *cell_list* and a buffer or a pair of inverters at the specified pins, or the net on a mapped design. When a driver pin or a net is given, a buffer or a pair of buffers is removed with this driver as the root.

This command does not remove buffer/inverter-pair across the logical hierarchy.

This command support transparent interface optimization(TIO) flow, which can remove buffer cells in sub blocks of abstract view from top block.

This command can remove inverters only as pairs. You must specify both of the pairing inverters if you use the *cell_list* argument.

This command accepts the final implementation of buffer or inverters removal, even if it negatively impacts the timing or Design Rule Checking (DRC) violations in the design.

This command support editing a verilog module in the module aspect, through **edit_module**.

This command will check the editability of the block before removing buffer cells.

This command honors design.eco_freeze_silicon_mode. When this app option is true, buffers to remove will be marked as spare cell and renamed by adding _Spare post-fix, instead of being removed from database.

This command is exactly equivalent to **remove_buffers**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes a buffer cell:

```
prompt> remove_buffer -from [get_pins cell1/O]
1

prompt> remove_buffer -net [get_nets net1]
1

prompt> remove_buffer cell2
1
```

Editing a verilog module in the module aspect:

```
prompt> edit_module h1m {remove_buffer {u2 u3 u4 u5 u6}}
```

SEE ALSO

remove_buffers(2)
all_fanout(2)

add_buffer(2)
report_cell(2)
edit_module(2)

remove_buffer_trees

Remove all buffer trees or buffer trees between specified drivers / loads

SYNTAX

status **remove_buffer_trees**
[-from *pin_or_net_list*]
[-source_of *pin_list*]
[-all]
[-verbose]

Data Types

pin_or_net_list N/A
pin_list N/A

ARGUMENTS

-from *pin_or_net_list*

Specifies a list of driver pins or nets from which buffer trees are to be removed. The command removes all buffers and inverters from these specified pins or nets until encountering the next non-repeaters loads. This option is mutually exclusive with the **-all** option. If this option is specified with **-source_of**, then buffer trees between the **-from** driver pins and **-source_of** load pins are removed.

-source_of *pin_list*

Specifies a list of load pins. The command removes all buffers and inverters between the load pin and the previous non-repeater driver pin, tracing upstream from the load. Only the buffers and inverters directly between that driver and the specified load pins are removed. There might be repeaters in the tree going only to other loads, and they will not be removed. If the cell of a specified pin is a repeater, the cell itself is also removed. This option is mutually exclusive with the **-all** option.

-all

Removes all buffer trees in the design. This option is mutually exclusive with the **-from** and **-source_of** option.

-verbose

Prints detailed info during buffer tree removal, such as buffer instance names whose removal introduce logical feedthroughs in the netlist.

DESCRIPTION

This command removes data path buffer trees in the design.

There are three options for removing buffer trees. It is required to specify one of these options:

- **-all**: Removes all buffer trees
- **-from**: Removes buffer trees from the specified drivers pins or nets
- **-source_of**: Removes the partial buffer trees in the fan-in to the specified load pins

Buffer tree removal is not performed on ideal nets, clock nets, tri-state nets, and multi-driven nets. Some buffers and inverters cannot be removed, including dont_touch cells, size_only cells, and unpaired inverters.

If a buffer removal would introduce a logical feedthrough in the netlist, and the app option **opt.port.eliminate_verilog_assign** is set to *true*, then that buffer will not be removed.

Use **remove_clock_trees** to remove clock trees.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes a buffer tree driven by pin a/O

```
prompt> remove_buffer_trees -from [get_pins a/O]
```

The following example removes the fan-in buffer tree driving pin b/I

```
prompt> remove_buffer_trees -source_of [get_pins b/I]
```

The following example removes all buffer trees

```
prompt> remove_buffer_trees -all
```

SEE ALSO

create_buffer_trees(2)
remove_clock_trees(2)
report_buffer_trees(2)
opt.port.eliminate_verilog_assign(3)

remove_buffers

Removes the buffer cells at a specified driver pin or net on a mapped design.

SYNTAX

status **remove_buffers**
-from *start_point*
-net *net_list*
cell_list
[-allow_tie_connection]

Data Types

start_point	list
net_list	list
cell_list	list
allow_tie_connection	boolean

ARGUMENTS

-from *start_point*

Specifies the starting pin or port of the fanout tree from which the buffers or inverter pairs need to be removed. Only driver pins or driver ports can be specified for this option.

This option is mutually exclusive with the **-net** and **cell_list** options.

-net *net_list*

Specifies the starting net of the fanout tree from which the buffers or inverter pairs need to be removed. Only one net can be specified for this option and the driver pin of this net would be considered as the start for fanout traversal. This means that all of the load cells connected to the net are considered in the buffer tree.

This option is mutually exclusive with the **-from** and **cell_list** options.

cell_list

Specifies the buffer or inverter cells to be removed. Each buffer is a cell name or a collection of netlist cells. When inverter cells are specified, they must be specified in pairs.

This option is mutually exclusive with **-from** options.

-allow_tie_connection

Specifies whether the buffers connected with the tie/PG net can be removed.

DESCRIPTION

This command removes buffer cells when you specify a *cell_list* and a buffer or a pair of inverters at the specified pins, or the net on a mapped design. When a driver pin or a net is given, a buffer or a pair of buffers is removed with this driver as the root.

This command does not remove buffer/inverter-pair across the logical hierarchy.

This command support transparent interface optimization(TIO) flow, which can remove buffer cells in sub blocks of abstract view from top block.

This command can remove inverters only as pairs. You must specify both of the pairing inverters if you use the *cell_list* argument.

This command accepts the final implementation of buffer or inverters removal, even if it negatively impacts the timing or Design Rule Checking (DRC) violations in the design.

This command support editing a verilog module in the module aspect, through **edit_module**.

This command will check the editability of the block before removing buffer cells.

This command honors design.eco_freeze_silicon_mode. When this app option is true, buffers to remove will be marked as spare cell and renamed by adding _Spare post-fix, instead of being removed from database.

This command is exactly equivalent to **remove_buffer**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes a buffer cell:

```
prompt> remove_buffers -from [get_pins cell1/O]
1

prompt> remove_buffers -net [get_nets net1]
1

prompt> remove_buffers cell2
1
```

Editing a verilog module in the module aspect:

```
prompt> edit_module h1m {remove_buffers {u2 u3 u4 u5 u6}}
```

SEE ALSO

remove_buffer(2)
all_fanout(2)

add_buffer(2)
report_cell(2)
edit_module(2)

remove_bundle_pin_constraints

Removes all bundle pin constraints from the bundles. If no options are given, all bundle pin constraints in the design are removed.

SYNTAX

int **remove_bundle_pin_constraints**
[-bundles *bundle_collection*]
[-cells *cell_collection*]
[-self]

Data Types

bundle_collection collection
cell_collection collection

ARGUMENTS

-bundles *bundle_collection*

Specifies the net bundles from which the constraints are removed. If not specified, it applies to all bundles. The bundles must be in the top level of the current design.

-cells *cell_collection*

Specifies the block cells from which the constraints are removed. If not specified, and **-self** is also not specified, the command processes all block cells. The cells must be in the top level of the current design.

-self

Specifies that the constraints applied to the top-level block are removed. If not specified, and **-cells** is also not specified, it applies to all block cells. It is legal to use **-cells** and **-self** together.

DESCRIPTION

This command removes all existing bundle pin constraints from nets, pins and ports. If you do not specify any arguments, this command removes *all* bundle pin constraints from the design. Note that you can specify both a set of bundles with **-bundles** and a set of block cells with **-cells** (or *-self*). In this case, only constraints which apply to both a specified bundle and a specified cell will be removed.

The **set_editability** command can enable or disable the **remove_bundle_pin_constraints** command for specified blocks or levels of physical hierarchy. This command has no effect on the blocks that are disabled with the **set_editability** command.

There are four type of bundle pin constraints, ordered from less specific to more specific:

- Global constraints: Constraint applies to all bundles and all block instances.
- Per-bundle constraints: Constraint applies to a specific bundle and all block instances.
- Per-cell constraints: Constraint applies to all bundles and a specific block instance.
- Per-bundle-cell-pair constraints: Constraint applies to a specific bundle and a specific block instance.

In general, this command removes the specified constraint and any other related, more specific constraint. For example, the command **remove_bundle_pin_constraints -bundles B** removes the per-bundle constraint specific to bundle B, but also removes the per-bundle-cell-pair constraints which are specific to bundle B and any cell related to bundle B. Constraints specific to only to a cell but not to bundle B are not removed. In addition, global constraints and are not removed in this example.

This command returns 1 if some constraints could be removed, 0 otherwise.

EXAMPLES

The following example removes all bundle pin constraints from the design, including global constraints, per-bundle constraints, per-cell constraints, and per-bundle-cell-pair constraints.

```
prompt> remove_bundle_pin_constraints
```

The following example removes all bundle pin constraints specific to BUNDLE0. Per-bundle and per-bundle-cell-pair constraints which are specific to BUNDLE0 are removed.

```
prompt> remove_bundle_pin_constraints -bundles [get_bundles BUNDLE0]
```

The following example removes all bundle pin constraints specific to cmem0. Per-cell and per-bundle-cell-pair constraints which are specific to cmem0 are removed.

```
prompt> remove_bundle_pin_constraints -cells [get_cells cmem0]
```

The following example removes all bundle pin constraints specific to the top-level block. Per-cell and per-bundle-cell-pair constraints which are specific to the top-level block are removed.

```
prompt> remove_bundle_pin_constraints -self
```

The following example removes all bundle pin constraints specific to the combination of BUNDLE0 and cmem0. Only matching per-bundle-cell-pair constraints are removed.

```
prompt> remove_bundle_pin_constraints -bundles [get_bundles BUNDLE0] \
  -cells [get_cells cmem0]
`.in -0.25in
```

SEE ALSO

place_pins(2)
remove_block_pin_constraints(2)
remove_individual_pin_constraints(2)
report_bundle_pin_constraints(2)
set_bundle_pin_constraints(2)

set_editability(2)

remove_bundles

Remove the specified bundle(s)

SYNTAX

status_value **remove_bundles**
bundle_list

bundle_list list

ARGUMENTS

bundle_list

Specifies a list of bundles to remove.

DESCRIPTION

Removes a list of bundles from the current design.

EXAMPLES

The following example removes bundles from the design

```
prompt> remove_bundles [get_bundles BUNDLE_5]
1
prompt> remove_bundles my_bundle
1
```

SEE ALSO

add_to_bundle(2)
create_bundle(2)
get_bundles(2)

remove_from_bundle(2)
report_bundles(2)

remove_busplans

Remove entire busplans or specified bits of busplans from design.

SYNTAX

int **remove_busplans**
[-from *pins_or_ports*]
[buses]

Data Types

pins_or_ports collection of pin or port objects
buses string

ARGUMENTS

-from *pins_or_ports*

Specify the busplans to be removed based on where the busplan begins. This allows you to remove a subset of the bits of a busplan. This is mutually exclusive with *buses*.

buses

Specify the busplans to remove based on the busplan names. The busplans must have been previously created by the **create_busplans** command. This is mutually exclusive with *-from*.

DESCRIPTION

Remove busplans that were previously defined by the **create_busplans** command.

EXAMPLES

Create a 8 bit busplan named bus1 and then remove one bit from that busplan.

prompt> **create_busplans -name bus1 -from B1/OUT1[*] -start_end_cells {B1 B2 B3 B4}**
prompt> **report_busplans bus1**

****

```
Report : report_busplans
 ...

****************************************
Start/end at instances: B1 B2 B3 B4
bus1: 8 bits start=B1/OUT1[7] clock=-unknown-
  pin              group1     (  8) B1/OUT1[7]
    register         group2     (  8) r1/r7
      register       group3     (  8) r2/r7
        pin          group4     (  8) B2/IN1[7]
1
prompt> remove_busplans -from B1/OUT1[7]
1
prompt> report_busplans bus1

****************************************
Report : report_busplans
 ...

****************************************
Start/end at instances: B1 B2 B3 B4
bus1: 7 bits start=B1/OUT1[6] clock=-unknown-
  pin              group1     (  7) B1/OUT1[6]
    register         group2     (  7) r1/r6
      register       group3     (  7) r2/r6
        pin          group4     (  7) B2/IN1[6]
1
```

SEE ALSO

create_busplans(2)
set_net_estimation_rule(2)

remove_case_analysis

Removes the case analysis value on the specified input.

SYNTAX

status **remove_case_analysis**
port_or_pin_list

Data Types

port_or_pin_list list

ARGUMENTS

port_or_pin_list

Lists ports or pins for which to remove the case analysis entry.

-all

Specifies to remove all case analysis in the current design.

DESCRIPTION

This command removes previously specified case analysis entries on ports or pins. Case analysis is specified using the **set_case_analysis** command. The command specifies that a pin or port is at a constant logic value (0 or 1), or that only one of the rising or falling transition is valid.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example specifies that ports in0 and in2 are set to the constant logic value 0.

```
prompt> set_case_analysis 0 {in0 in2}
prompt> report_case_analysis
```

```
****************************************
Report : case_analysis
 ...

****************************************

Pin name     Case analysis value
---------------------------------
in2          0
in0          0
```

The following example removes the case analysis entry on port in2.

```
prompt> remove_case_analysis {in2}
prompt> report_case_analysis

****************************************
Report : case_analysis
Design : middle
Version:
Date   :
****************************************

Pin name     Case analysis value
---------------------------------
in0          0
```

The following example removes all specified mode that has been set from the current design.

```
prompt> remove_case_analysis -all
```

SEE ALSO

set_case_analysis(2)
report_case_analysis(2)

remove_cell

Removes the specified cell instances.

SYNTAX

integer **remove_cells**
[-design *design*]
[-force]
[*cells*]
[-all]

Data Types

design collection
cells collection

ARGUMENTS

-design *design*

Specifies the block from which to remove the cells.

By default, the cells are removed from the current block.

-force

Removes the specified cell instances even if their **physical_status** attribute is set to **locked**.

By default, the command raises a Tcl error if the **physical_status** attribute of any of the specified cells has a value of **locked**.

cells

Specifies the cell instances to remove.

This argument and the **-all** option are mutually exclusive; you must specify one.

-all

Removes all cell instances from the specified block.

This option and the *cells* argument are mutually exclusive; you must specify one.

DESCRIPTION

Removes the specified cell instances from the current block unless otherwise specified by the **-design** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all cells whose instance names start with "or2t".

```
prompt> get_cells or2t*
{or2t_1 or2t_2}

prompt> remove_cells or2t*
2
```

SEE ALSO

create_cell(2)
get_cells(2)
replace_cell(2)

remove_cell_array_patterns

Removes cell array patterns.

SYNTAX

int **remove_cell_array_patterns**
[-all]
[-verbose]
[*cell_array_pattern_list*]

Data Types

cell_array_pattern_list list

ARGUMENTS

-all

Removes all cell_array_patterns from the current block.

-verbose

Displays verbose deletion information.

cell_array_pattern_list

Specifies a list of cell array patterns to remove. The list may contain cell array pattern names, patterns, or collections. A collection may be specified by using the **get_cell_array_patterns** command.

RETURN VALUE

The number of removed cell array patterns.

DESCRIPTION

This command removes cell_array_patterns from the current block.

EXAMPLES

The following example removes cell_array_patterns that match the pattern "CELL_ARRAY_PATTERN*".

```
prompt> remove_cell_array_patterns CELL_ARRAY_PATTERN*
3
```

The following example removes all cell_array_patterns in the block.

```
prompt> remove_cell_array_patterns -all
5
```

SEE ALSO

get_cell_array_patterns(2)
create_cell_array_pattern(2)
report_cell_array_patterns(2)

remove_cell_buses

Removes cell buses from the current design.

SYNTAX

collection **remove_cell_buses**
[-design *design*]
[-force]
[-block *block*]
[-cell *cell*]
-all | *cell_bus_list*

Data Types

block string or collection
cell string or collection
cell_bus_list string or collection

ARGUMENTS

-design *design*

Specifies the design in which to remove the cell buses. If no design is specified, the cell buses are removed from the current design.

-block *block*

Specifies the block where the cell bus is to be removed. If specified, **remove_cell_buses** has the same effect as **edit_module** in which the module is changed and changes will be on all module occurrences.

-cell *cell*

Specifies the cell where the cell bus is to be removed. The cell bus is removed on the cell's reference module. **remove_cell_buses** will first uniquify the reference if needed, then remove the cell bus. When not specified, the cell bus is removed on the current module.

-force

Ignores the locked status of cells of the bus cell. If this option is not specified and any of the affected objects has locked status, the command issues a Tcl error and exits.

-all

Removes all cell buses.

cell_bus_list

Specifies a list of cell buses to remove.

RETURN VALUE

This command returns the removed cell bus count, an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

DESCRIPTION

This command removes a list of cell buses from the current design (unless otherwise specified by -design). Individual members of buses remain.

EXAMPLES

The following example removes cell bus named *bus1*

```
prompt> remove_cell_buses bus1
1
```

SEE ALSO

create_cell_bus(2)
get_cell_buses(2)
report_cell_buses(2)

remove_cells

Removes the specified cell instances.

SYNTAX

integer **remove_cells**
[-design *design*]
[-force]
[*cells*]
[-all]

Data Types

design collection
cells collection

ARGUMENTS

-design *design*

Specifies the block from which to remove the cells.

By default, the cells are removed from the current block.

-force

Removes the specified cell instances even if their **physical_status** attribute is set to **locked**.

By default, the command raises a Tcl error if the **physical_status** attribute of any of the specified cells has a value of **locked**.

cells

Specifies the cell instances to remove.

This argument and the **-all** option are mutually exclusive; you must specify one.

-all

Removes all cell instances from the specified block.

This option and the *cells* argument are mutually exclusive; you must specify one.

DESCRIPTION

Removes the specified cell instances from the current block unless otherwise specified by the **-design** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all cells whose instance names start with "or2t".

```
prompt> get_cells or2t*
{or2t_1 or2t_2}

prompt> remove_cells or2t*
2
```

SEE ALSO

create_cell(2)
get_cells(2)
replace_cell(2)

remove_checkpoint_actions

Removes the specified action's definitions, associations, and history from the checkpoint system.

SYNTAX

integer **remove_checkpoint_actions**
action_names *action_names*

Data Types

action_names list

ARGUMENTS

-action_names *action_names*

Specifies the names of the action or flow change for the checkpoint system that will be removed.

DESCRIPTION

This command removes the specified actions from the checkpoint system. Removal includes the action definition created by create_checkpoint_action, the associations to checkpoints defined by associate_checkpoint_action, and any history of that action having run at previous checkpoints reported by get_checkpoint_data.

EXAMPLES

The following example removes a checkpoint action or flow change for the checkpoint system.

This example removes the high_congestion_effort action.

```
prompt> remove_checkpoint_actions high_congestion_effort
```

This example removes the power_mode_total and power_mode_leakage actions.

```
prompt> remove_checkpoint_actions "power_mode_total power_mode leakage"
```

SEE ALSO

create_checkpoint_action(2)
create_checkpoint_report(2)
remove_checkpoint_reports(2)
associate_checkpoint_action(2)
associate_checkpoint_report(2)
eval_checkpoint(2)
get_current_checkpoint(2)
set_checkpoint_options(2)
reset_checkpoints(2)
get_checkpoint_data(2)

remove_checkpoint_reports

Removes the specified report's definitions, associations, and history from the checkpoint system.

SYNTAX

integer **remove_checkpoint_reports**
  report_names *report_names*

Data Types

report_names list

ARGUMENTS

-report_names *report_names*

Specifies the names of the report for the checkpoint system that will be removed.

DESCRIPTION

This command removes the specified reports from the checkpoint system. Removal includes the report definition created by create_checkpoint_report, the associations to checkpoints defined by associate_checkpoint_report, and any history of that report having run at previous checkpoints reported by get_checkpoint_data.

EXAMPLES

This example removes the timing report.

```
prompt> remove_checkpoint_actions timing
```

This example removes the area and power reports.

```
prompt> remove_checkpoint_actions "area power"
```

SEE ALSO

create_checkpoint_action(2)
create_checkpoint_report(2)
remove_checkpoint_actions(2)
associate_checkpoint_action(2)
associate_checkpoint_report(2)
eval_checkpoint(2)
get_current_checkpoint(2)
set_checkpoint_options(2)
reset_checkpoints(2)
get_checkpoint_data(2)

remove_clock

Removes one or more clocks from the current design.

SYNTAX

string **remove_clocks** -all | *clock_list*

list *clock_list*

ARGUMENTS

-all

Specifies to remove all clocks in the current design.

clock_list

Specifies a list of collections containing clocks or patterns matching the clock names.

DESCRIPTION

Removes the specified clock objects from the current design. You must specify either *clock_list* or **-all**. The command returns the number of clocks removed.

If a removed clock is the only member of a path group, the path group is also removed. For information about path groups, refer to the **group_path** man page. To list all clock sources in the design, use **report_clock**.

All arrival and required times specified by commands **set_input_delay** and **set_output_delay** relatively to the clock that is being removed are also removed.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example removes clock CLK1.

prompt> **remove_clocks CLK1**

The following example removes all clocks from the current design.

prompt> **remove_clocks -all**

SEE ALSO

create_clock(2)
current_design(2)
get_clocks(2)
group_path(2)
report_clock(2)
set_input_delay(2)
set_output_delay(2)
reset_design(2)

remove_clock_balance_groups

Remove clock balance groups.

SYNTAX

string **remove_clock_balance_groups**
clock_balance_group_list | -all

Data Types

clock_balance_group_list collection

ARGUMENTS

clock_balance_group_list

Specifies a list of clock balance groups to remove. This option and **-all** option are mutually exclusive.

-all

Remove all clock balance groups in the current design. This option and *clock_balance_group_list* are mutually exclusive. Either the **-all** option or *clock_balance_group_list* has to be specified.

DESCRIPTION

This command will remove clock balance groups from constraints.

Multicorner-Multimode Support

EXAMPLES

The following example removes the specified clock_balance_group.

prompt> **remove_clock_balance_groups grp1**

SEE ALSO

create_clock_balance_group(2)
report_clock_balance_groups(2)
get_clock_balance_groups(2)
balance_clock_groups(2)

remove_clock_balance_points

Removes user-specified constraints set with a previous **set_clock_balance_points** command.

SYNTAX

string **remove_clock_balance_points**
[-clock *clock_list*]
[-corners *corner_list*]
[-balance_points *port_pin_list*]

Data Types

clock_list list
corner_list list
port_pin_list list

ARGUMENTS

-clock *clock_list*

Specifies the clocks from which to remove the constraints.

By default, the command removes the constraints from all clocks of the current mode.

-corners *corner_list*

Specifies the corners from which to remove the constraints.

By default, the command removes the constraints from all corners of the current mode.

-balance_points *port_pin_list*

Specifies the ports or pins from which to remove the constraints.

By default, the command removes all constraints from the specified clocks.

DESCRIPTION

This command removes user-specified constraints set with a previous **set_clock_balance_points** command.

If the command does not find any constraints to remove, it issues an information message.

EXAMPLES

The following example removes the constraint set on the gck/CP pin for all clocks in the current mode.

```
prompt> remove_clock_balance_points -balance_points gck/CP
```

SEE ALSO

set_clock_balance_points(2)
report_clock_balance_points(2)

remove_clock_cell_spacings

Removes spacing rules for the specified or all cells in the clock tree.

SYNTAX

status **remove_clock_cell_spacings**
[-clocks *clock_list*]
[-lib_cells *libcell_name_list*]

Data Types

clock_list collection
libcell_name_list list

ARGUMENTS

-clocks *clock_list*

Specifies the clock trees for which the cell spacing settings will be removed.

-lib_cells *libcell_name_list*

Specifies a list of library cell names, so that only instances on the clock tree of such type would be affected.

DESCRIPTION

This command serves as an instruction to mark relevant existing clock cell spacing rules, previously defined by **set_clock_cell_spacing_rule** command for the removal. In order for the rules to be removed from the relevant instances on the clock network, **synthesize_clock_trees** command needs to be executed. If **-lib_cells** switch is not specified all spacing rules previously defined would be marked for the removal. If it (**-lib_cells** switch) is defined - only spacing rules, that are applied to specified libcells would be marked for the removal. If **-clocks** option is used then it will remove cell spacing settings for specified clock networks.

EXAMPLES

The following example marks clock cell spacing rules for the removal for all clock buffers that are of the cell type BUFFD2BWP

prompt> **remove_clock_cell_spacings -lib_cells mylib/BUFFD2BWP**

The following example marks all clock cell spacing rules for the removal.

```
prompt> remove_clock_cell_spacings
```

SEE ALSO

report_clock_cell_spacings(2)
set_clock_cell_spacing(2)
cts.placement.cell_spacing_rule_style(3)

remove_clock_drivers

Removes cells added with command *create_clock_drivers*.

SYNTAX

status **remove_clock_drivers**
[-prefix *name*]

Data Types

name string

ARGUMENTS

-prefix *name*

Only those cells are deleted which have a name starting exactly with the specified prefix. The passed name should not contain a wildcard.

DESCRIPTION

The *remove_clock_drivers* command will delete cells inserted by previous calls of command *create_clock_drivers*. It will remove all repeaters and merge template clones into the originally given template cell. Merging of template clones will only be done if every equivalent input is driven by the same pin as the equivalent original template cell input pin. With the option *-prefix* the set of removed cells can be limited to those which start exactly with the specified prefix.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example removes all cells inserted with calls to *create_clock_drivers* having a name starting with foo.

prompt> **remove_clock_drivers -prefix foo**

SEE ALSO

create_clock_drivers(2)
synthesize_multisource_global_clock_trees(2)
remove_multisource_global_clock_trees(2)

remove_clock_exclusivity

Removes clock exclusivity setting on a cell output set by set_clock_exclusivity command.

SYNTAX

Boolean **remove_clock_exclusivity**
-output output_pins

output_pins *list*

ARGUMENTS

-output output_pins

Specifies the output pin of a cell that has been previously set as exclusive by the set_clock_exclusivity command.

DESCRIPTION

This command removes the exclusivity previously set on an output pin of a cell by the set_clock_exclusivity command. In that case, the tool will no longer infer exclusivity of clocks that pass through the cell and reach that output pin.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example removes two exclusivity points previously set by set_clock_exclusivity command.

prompt> **remove_clock_exclusivity -output {AND02/Z MUX02/Z}**

SEE ALSO

set_clock_exclusivity(2)

report_clock(2)
set_disable_auto_mux_clock_exclusivity(2)
time.enable_auto_mux_clock_exclusivity(3)

remove_clock_gating_check

Removes clock gating checks for design objects.

SYNTAX

status **remove_clock_gating_check**
[-setup]
[-hold]
[-rise]
[-fall]
[-high | -low]
[*object_list*]

Data Types

object_list list

ARGUMENTS

-setup

Removes the clock-gating constraint on the setup time only. If you do not specify either the **-setup** or **-hold** option, both setup and hold constraints are removed.

-hold

Removes the clock-gating constraint on the hold time only. If you do not specify either the **-setup** or **-hold** option, both setup and hold constraints are removed.

-rise

Removes the clock-gating constraint only on the rising delays. If you do not specify either the **-rise** or **-fall** option, constraints on both rising and falling delays are removed.

-fall

Removes the clock-gating constraint only on the falling delays. If you do not specify either the **-rise** nor **-fall** option, constraints on both rising and falling delays are removed.

-high

Removes the high specification from the object list, previously set by the **set_clock_gating_check** command. You must specify either **-high** or **-low**.

-low

Removes the low specification from the object list, previously set up by **set_clock_gating_check** command. This option has to be either high or low.

object_list

Specifies a list of objects in the current design for which to remove the clock gating check. The objects can be clocks, ports, pins, or cells. If you specify a cell, all input pins of that cell are affected. If you do not specify any objects, the clock-gating check is removed from the current design.

DESCRIPTION

The **remove_clock_gating_check** command is used to remove any clock-gating check properties applied with the **set_clock_gating_check** command. This command does not remove the clock-gating check itself.

Multicorner-Multimode Support

EXAMPLES

The following example removes the setup requirement (for rising and falling delays) on all gates in the clock network involved with clock CK1 path.

```
prompt> remove_clock_gating_check -setup [get_clocks CK1]
```

The following example removes the hold requirement on the rising delay of gate and1.

```
prompt> remove_clock_gating_check -hold -rise [get_cells and1]
```

You can also use the **reset_design** command to remove information set by the **set_clock_gating_check** command.

SEE ALSO

current_design(2)
remove_disable_clock_gating_check(2)
reset_design(2)
set_clock_gating_check(2)
set_disable_clock_gating_check(2)
time.disable_clock_gating_checks(3)
time.clock_gating_user_setting_only(3)

remove_clock_groups

Removes specific exclusive or asynchronous clock groups from the current design.

SYNTAX

Boolean **remove_clock_groups**
-physically_exclusive | -logically_exclusive | -asynchronous
-name *name_list* | -all

list *name_list*

ARGUMENTS

-physically_exclusive

Specifies that groups set for physically exclusive clocks are to be removed. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-logically_exclusive

Specifies that groups set for logically exclusive clocks are to be removed. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-asynchronous

Specifies that groups set for asynchronous clocks are to be removed. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-name *name_list*

Specifies a list of clock groups to be removed, which matches the groups in the given names. You should use the **set_clock_groups** command to predefine these names. Substitute the list you want for *name_list*. The **-name** and **-all** options are mutually exclusive.

-all

Specifies to remove all groups set for exclusive or asynchronous clocks in the current design. The **-name** and **-all** options are mutually exclusive.

DESCRIPTION

Removes specific exclusive or asynchronous clock groups from the current design. These clock groups are specified by the

name_list from the current design. You must specify either **-exclusive** or **-asynchronous**. You must specify either *name_list* or **-all**.

To find the names for existing groups, use the **report_clock** command with the **-groups** option.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example removes exclusive clock groups named *mux*.

```
prompt> remove_clock -exclusive -name mux
```

The following example removes all asynchronous clock groups from the current design.

```
prompt> remove_clock -asynchronous -all
```

SEE ALSO

report_clock(2)
set_clock_groups(2)

remove_clock_jitter

Removes clock jitter information previously set by the **set_clock_jitter** command.

SYNTAX

status **remove_clock_jitter**
-clock *clock_list*
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]

Data Types

clock_list list

ARGUMENTS

-clock *clock_list*

Specifies a list of clocks from which to remove the clock jitter.

-modes *mode_list*

Specifies the scenarios to remove the clock jitter. If this option is given, all scenarios of the given modes are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-corners *corner_list*

Specifies the scenarios to remove the clock jitter. If this option is given, all scenarios of the given corners and the current mode is used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-scenarios *scenario_list*

Specifies the scenarios to remove the clock jitter. The **-modes** or **-corners** option must not be given with this option.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

DESCRIPTION

This command removes the cycle-to-cycle jitter or the duty-cycle jitter of the clock that was previously set by the **set_clock_jitter** command.

EXAMPLES

The following example removes the cycle-to-cycle jitter of 0.5 and duty-cycle jitter of 0.7 specified on the clock mclk

```
prompt> set_clock_jitter -clock [get_clocks mclk] -cycle 0.5 -duty_cycle 0.7
prompt> remove_clock_jitter -clock [get_clocks mclk]
```

SEE ALSO

report_clock_jitter(2)
set_clock_jitter(2)

remove_clock_latency

Removes clock latency information from specified objects.

SYNTAX

string **remove_clock_latency**
[-source]
[-offset]
[-clock *clock_list*]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
object_list

Data Types

clock_list list
mode_list list
corner_list list
scenario_list list
object_list list

ARGUMENTS

-source

Specifies that clock source latency should be removed.

-offset

Specifies that clock offset latency should be removed.

-clock *clock_list*

Removes from the design any network latency defined on the pin or port objects in *object_list* which refers the clocks in *clock_list*. If the **-clock** option is supplied when *object_list* refers to clock objects, a warning is issued that the option is not relevant in this case and execution of the command proceeds as if **-clock** was not specified. This option does not remove a more general latency setting without any specific clock.

-modes *mode_list*

Specifies the scenarios from which to remove the clock latency. If this option is given, all scenarios of the specified modes will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners *corner_list*

Specifies the scenarios from which to remove the clock latency. If this option is given, all scenarios of the specified corners and the current mode will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios ***scenario_list***

Specifies the scenarios from which to remove the clock latency. The **-modes** or **-corners** option must not be specified with this option.

object_list

Provides a list of clocks, ports, or pins.

DESCRIPTION

This command removes clock latency information from the specified objects. It removes the user-specified clock network or source latency information from the specified objects. If a dynamic component of clock source latency has been specified, this will also be removed. If separate latency values for corners exist (specified using **-corners** in the **set_clock_latency** command) then all those corner values are also removed. Clock network latency is the time required for a clock signal on the design to propagate from the clock definition point to a register clock pin. Clock source latency (also called insertion delay) is the time it takes for a clock signal to propagate from its actual ideal waveform origin point to the clock definition point in the design.

Clock latency is managed on a per-scenario basis. For designs with multiple scenarios, you can specify scenarios by using the **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command applies to the current scenario. The tool issues an error if there is no current scenario. If the **-scenarios** option is specified, the command applies to all of the specified scenarios.

If the **-modes** option is specified, the command applies to all scenarios of the specified modes. The tool issues an error if none of the specified modes have any scenarios.

If the **-corners** option is specified, the command applies to all scenarios of the specified corners and the current mode. The tool issues an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are specified, the command applies to all scenarios of the specified corners and the specified modes. The tool issues an error if there are no such scenarios. It is an error to specify **-scenarios** with **-modes** or **-corners**.

Clock network and source latency information is set on objects by using the **set_clock_latency** command. See the **set_clock_latency** man page for more details.

To report clock network and source latency information, use the **report_clocks** command with the **-skew** option.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes clock latency information from the clock named **CLK1**.

```
prompt> remove_clock_latency [get_clocks CLK1]
```

The following example removes clock source latency information from the clock named **CLK1**.

```
prompt> remove_clock_latency -source \
[get_clocks CLK1]
```

The following example removes clock source latency information from the clock source port, CLK, with relative clock, **CLK1**. Dynamic component with the source latency is also removed.

```
prompt> remove_clock_latency -source \
-clock [get_clocks CLK1] [get_ports CLK]
```

SEE ALSO

create_clock(2)
remove_clock_uncertainty(2)
remove_clocks(2)
report_clocks(2)
set_clock_latency(2)
set_clock_uncertainty(2)

remove_clock_routing_rules

Removes routing rules in clock tree synthesis.

SYNTAX

status **remove_clock_routing_rules**
[-clocks *clock_list* | -nets *net_list*]
[-net_type sink | internal | root]
[-rule *ndr_rule* | -default_rule]

Data Types

clock_list collection
net_list collection
ndr_rule string

ARGUMENTS

-clocks *clock_list*

Specifies the clock trees of the rules . This option and the **-nets** option are mutually exclusive.

-nets *net_list*

Specifies the nets on which the routing rules are removed. This option and the **-clocks** option are mutually exclusive.

-net_type sink | internal | root

Specifies the net type of the routing rules. This option cannot work together with the **-nets** option. There are three net types: **sink**, **internal** and **root**.

-rule *ndr_rule*

Specifies the non-default routing rule to be removed.

-default_rule

Removes the default routing rule.

DESCRIPTION

This command removes the routing rules used in clock tree synthesis.

EXAMPLES

The following example assigns a non-default rule named *dblespacing* to the sink level nets of clock clk1 in clock tree. Then it removes this rule and reports the routing rules to confirm the settings.

```
prompt> set_clock_routing_rules -clocks clk1 -net_type sink -rule dblespacing
1
prompt> report_clock_routing_rules

****************************************
 Report : clock routing rules
 Design : Decoder
 Date   : Thu Nov 29 08:44:45 2012
****************************************
   Clock: clk1 (mode default); Net Type: sink;     Rules: dblespacing

prompt> remove_clock_routing_rules -clocks clk1 -net_type sink
1
prompt> report_clock_routing_rules

****************************************
 Report : clock routing rules
 Design : Decoder
 Date   : Thru Nov 29 08:45:12 2012
****************************************

 No clock routing rule is found.
```

SEE ALSO

set_clock_routing_rules(2)
report_clock_routing_rules(2)

remove_clock_sense

The remove_clock_sense command is deprecated. Use the remove_sense command instead.

Multicorner-Multimode Support

This command works only on the current mode.

SEE ALSO

remove_sense(2)
set_sense(2)

remove_clock_skew_groups

Removes the user-defined skew group.

SYNTAX

remove_clock_skew_groups
skew_group_list

Data Types

skew_group_list collection

ARGUMENTS

skew_group_list

List of skew group names/collections.

DESCRIPTION

This command removes the specified skew group from the design.

Multicorner-Multimode Support

EXAMPLES

The following example removes the skew group named grp1.

```
prompt> remove_clock_skew_groups [get_clock_skew_groups grp1]
in -0.25i
```

SEE ALSO

create_clock_skew_group(2)

get_clock_skew_group(2)
report_clock_skew_groups(2)

remove_clock_transition

Removes clock transition time information.

SYNTAX

string **remove_clock_transition** *clock_list*

clock_list
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]

clock_list list
mode_list list
corner_list list
scenario_list list

ARGUMENTS

clock_list

Specifies a list of clocks for which to remove clock transition time.

-modes ***mode_list***

Specifies the scenarios that clock transitions will be removed from. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option may not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners ***corner_list***

Specifies the scenarios that clock transitions will be removed from. If this option is given, all scenarios of the given corners and the current mode will be used. The **-scenarios** option may not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios ***scenario_list***

Specifies the scenarios that clock transitions will be removed from. The **-modes** or **-corners** option may not be given with this option.

DESCRIPTION

Removes clock transition information specified by the **set_clock_transition** command. To list all the set clock transition values, use the **report_clock** command with the **-skew** option.

Clock transition is managed on a per-scenario basis. For designs with multiple scenarios, you can specify scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command is applied to the current scenario. (It will be an error if there is no current scenario). If the **-scenarios** option is given, the command will be applied to all of the specified scenarios. If the **-modes** option is given, the command will be applied to all scenarios of the specified modes. (It will be an error if none of the specified modes have any scenarios.) If the **-corners** option is given, the command will be applied to all scenarios of the specified corners and the current mode. (It will be an error if none of the specified corners have any scenarios with the current mode.) If both the **-modes** and **-corners** options are given, the command will be applied to all scenarios of the specified corners and the specified modes. (It will be an error there are no such scenarios.) It is an error to give **-scenarios** with **-modes** or **-corners**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes clock transition information from all clocks in the current scenario.

```
prompt> remove_clock_transition [all_clocks]
```

SEE ALSO

all_clocks(2)
create_clock(2)
report_clocks(2)
set_clock_latency(2)
set_clock_transition(2)
set_clock_uncertainty(2)

remove_clock_tree_options

Remove existing target skew or latency constraints.

SYNTAX

status **remove_clock_tree_options**
[-clocks *clock_list*]
[-corners *corner_list*]
[-all]
[-target_skew]
[-target_latency]
[-copy_exceptions_across_modes]
[-root_ndr_fanout_limit]
[-max_incr_repeater_levels]

Data Types

clock_list list *corner_list* list

ARGUMENTS

-clocks *clock_list*

Removes target skew or latency on specified clock trees. If not specified, target skew or latency that is previously defined by **set_clock_tree_options** without -clocks option will be removed.

-corners *corner_list*

Removes target skew or latency in specified corners. If not specified, target skew or latency that is set to current corner will be removed.

-all

Removes all target skew or latency constraints.

-target_skew

Specifies that the constraints to be removed are target skew constraints.

-target_latency

Specifies that the constraints to be removed are target latency constraints.

-copy_exceptions_across_modes

Specifies that all the "-copy_exceptions_across_modes" settings will be removed. It can only be used by itself and cannot be

combined with other options.

-root_ndr_fanout_limit

Specifies that "-root_ndr_fanout_limit" (fanout-based ndr) settings will be removed. It can be combined with all other options except the "copy_exceptions_across_modes" option.

-max_incr_repeater_levels

Specifies that "-max_incr_repeater_levels" settings will be removed.

DESCRIPTION

Remove target skew or latency constraints or fanout-based ndr settings that are previously defined by **set_clock_tree_options**.

EXAMPLES

The following example removes the target skew constraint on clock clk1.

prompt> **remove_clock_tree_options -target_skew -clock clk1**

The following example removes all target latency constraint on all clocks.

prompt> **remove_clock_tree_options -target_latency -all**

SEE ALSO

set_clock_tree_options(2)
report_clock_tree_options(2)

remove_clock_tree_reference_subset

Removes previously defined references from a clock tree.

SYNTAX

status **remove_clock_tree_reference_subset**
[-clocks *clock_list*]

Data Types

clock_list collection

ARGUMENTS

-clocks *clock_list*

Removes references settings for the specified list of clocks.

DESCRIPTION

Use this command to remove the specified references settings from clock trees. If the **-clocks** option is used, reference lists of the specified clocks are removed. If the **-clocks** option is not used, all clock tree reference settings will be removed.

EXAMPLES

The following example removes reference cell settings for clock tree clk1.

prompt> **remove_clock_tree_references_subset -clocks clk1**

SEE ALSO

report_clock_tree_reference_subset(2)
set_clock_tree_reference_subset(2)

remove_clock_trees

Removes buffers and inverters in the clock trees.

SYNTAX

status **remove_clock_trees**
[-clocks *clock_list*]
[-clock_repeaters_only]
[-keep_ideal_clock_attributes]
[-keep_struct]

Data Types

clock_list collection

ARGUMENTS

-clocks *clock_list*

Runs only the clock trees specified in *clock_list*. By default, the command is applied for all currently defined clock trees in all active modes.

-clock_repeaters_only

Removes buffers/inverters inserted by clock tree synthesis tools only. This option is off by default.

-keep_ideal_clock_attributes

Keeps the ideal clock attributes, such as clock latencies, clock transitions, and clock uncertainties. Those cells with ideal clock attributes will not be removed in this situation.

-keep_struct

This option will skip reparenting gates, so the clock tree's structure will not change.

DESCRIPTION

This command removes buffers and inverters in clock trees.

EXAMPLES

The following example removes all buffers in clock tree named CLK1.

```
prompt> remove_clock_trees -clocks CLK1
```

SEE ALSO

synthesize_clock_trees(2)

remove_clock_trunk_endpoints

Removes the status of clock trunk endpoint from pins or ports.

SYNTAX

status **remove_clock_trunk_endpoints**
[-clock *clock_list*]
[-corners *corner_list*]
[-auto_clock connected | local | all]
[*pins_or_ports*]

Data Types

clock_list list or collection
corner_list list or collection
pins_or_ports collection

ARGUMENTS

-clock *clock_list*

Removes clock trunk endpoints for the specified clocks. By default, clock trunk endpoints are removed for all clocks.

-corners *corner_list*

Removes corner trunk endpoints for the specified corners. By default, clock trunk endpoints are removed for all corners.

pins_or_ports

Removes clock trunk endpoints for the specified pins or ports. By default, clock trunk endpoints are removed for all pins and ports.

-auto_clock connected | local | all

Associates clocks at the block level with clocks at the top level. For example, your top-level clock might be called "SYS_CLK", and the same clock in the block might be called "CLK". You can set the association between block clock and top clock by using the **set_block_to_top_map** command, or by using **remove_clock_trunk_endpoints -auto_clock** and specifying the same rule. See the **set_block_to_top_map** man page for more details. Note that the **-clocks** and **-auto_clock** options are mutually exclusive. You can select one of three rules for auto-clock mapping:

- **connected** - The "connected" rule associates block-level clocks with top-level clocks under the following conditions: If a block clock has its source at a block port (a boundary clock), it can be associated to a top-level clock that propagates to that port. If a block clock has its source on an internal pin sources (a local clock), the top-level clock source must be declared at the same pin.
- **local** - The "local" rule maps connected clocks. In addition, if a block clock has its source on an internal pin sources (a local clock) and has no associated top-level clock, a new top-level clock will be created with the same waveform as the block-

level clock. The new top-level clock will be associated with the local block-level clock.

- **all** - The "all" rule maps clocks as in the "local" rule. In addition, if a block clock has its source at a block port (a boundary clock) and no top-level clock propagates to that port, a new top-level clock will be created with the same waveform as the block-level clock. The new top-level clock will be associated with the local block-level clock. The top-level boundary clock will allow timing analysis to be done, but it is not necessarily correct for signoff. You should consider removing or replacing this clock before signoff.

Note that if two block-level clocks are declared at the same boundary port, the "all" rule cannot resolve both of them. This is because it will not automatically declare two conflicting clocks in the top-level block. You must manually declare two top-level clocks, with the appropriate waveforms. After the top clocks exist, **-auto_clock** mapping can associate them with the block clocks.

DESCRIPTION

This command removes clock trunk endpoints during clock trunk planning. Use options to remove clock trunk endpoints only for specified clocks, corners, pins, or ports.

EXAMPLES

The following example removes all clock trunk endpoints.

```
prompt> remove_clock_trunk_endpoints
1
```

The following example removes clock trunk endpoints for the block1/CLK pin.

```
prompt> remove_clock_trunk_endpoints block1/CLK
1
```

SEE ALSO

gui_load_clock_trunk_planning(2)
report_clock_trunk_endpoints(2)
set_clock_trunk_endpoints(2)

remove_clock_uncertainty

Removes clock uncertainty information previously set by the **set_clock_uncertainty** command.

SYNTAX

string **remove_clock_uncertainty**
[*object_list* |
-from *from_clock*
| -rise_from *rise_from_clock*
| -fall_from *fall_from_clock*
-to *to_clock*
| -rise_to *rise_to_clock*
| -fall_to *fall_to_clock*]
[-rise]
[-fall]
[-setup]
[-hold]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[*object_list*]

Data Types

object_list list
from_clock list
rise_from_clock list
fall_from_clock list
to_clock list
rise_to_clock list
fall_to_clock list
mode_list list
corner_list list
scenario_list list

ARGUMENTS

-from *from_clock*

Specifies the source clock for interclock uncertainty. You must specify either the pair of **-from**/**-rise_from**/**-fall_from** and **-to**/**-rise_to**/**-fall_to**, or *object_list*; you cannot specify both.

-rise_from *rise_from_clock*

Same as the **-from** option, but indicates that *uncertainty* applies only to rising edge of the source clock. You can use only one of the **-from**, **-rise_from**, or **-fall_from** options. Use **-rise_from** instead of the obsolete option **-from_edge rise**.

-fall_from ***fall_from_clock***

Same as the **-from** option, but indicates that *uncertainty* applies only to the falling edge of the source clock. You can use only one of the **-from**, **-rise_from**, or **-fall_from** options. Use **-fall_from** instead of the obsolete option **-from_edge fall**.

-to ***to_clock***

Specifies the destination clock for interclock uncertainty. You must specify either the pair of **-from**/**-rise_from**/**-fall_from** and **-to**/**-rise_to**/**-fall_to**, or *object_list*; you cannot specify both.

-rise_to ***rise_to_clock***

Same as the **-to** option, but indicates that *uncertainty* applies only to rising edge of the destination clock. You can use only one of the **-to**, **-rise_to**, or **-fall_to** options. Use **-rise_to** instead of the obsolete option **-to_edge rise**.

-fall_to ***fall_to_clock***

Same as the **-to** option, but indicates that *uncertainty* applies only to falling edge of the destination clock. You can use only one of the **-to**, **-rise_to**, or **-fall_to** options. Use **-fall_to** instead of the obsolete option **-to_edge fall**.

-modes ***mode_list***

Specifies the scenarios that uncertainty will be set on. If this option is given, all scenarios of the given modes are used. The **-scenarios** option must not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-corners ***corner_list***

Specifies the scenarios that uncertainty will be set on. If this option is given, all scenarios of the given corners and the current mode are used. The **-scenarios** option may not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios ***scenario_list***

Specifies the scenarios that uncertainty will be set on. The **-modes** or **-corners** option must not be given with this option.

object_list

Specifies a list of clocks, ports or pins from which uncertainty information is to be removed. You can use either the pair of **-from**/**-rise_from**/**-fall_from** and **-to**/**-rise_to**/**-fall_to** options or the *object_list* option, but you cannot specify both; they are mutually exclusive.

-rise

Specifies that uncertainty is to be removed for only the rising clock edge. By default, uncertainty is removed for both rising and falling clock edges. This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use **-rise_to** instead.

-fall

Specifies that uncertainty is to be removed for only the falling clock edge. By default, uncertainty is removed for both rising and falling clock edges. This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use **-fall_to** instead.

-setup

Specifies that only setup check uncertainty is to be removed. By default, both setup and hold check uncertainties are removed.

-hold

Specifies that only hold check uncertainty is to be removed. By default, both setup and hold check uncertainties are removed.

DESCRIPTION

Removes clock uncertainty information previously set by the **set_clock_uncertainty** command from clocks, ports, pins, or between specified clocks. To display clock uncertainty information, use the **report_clocks** command with the **-skew** option.

Clock uncertainty is managed on a per-scenario basis. For designs with multiple scenarios, you can specify scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command is applied to the current scenario, and it is an error if there is no current scenario. If the **-scenarios** option is given, the command is applied to all of the specified scenarios. If the **-modes** option is given, the command is applied to all scenarios of the specified modes, and it is an error if none of the specified modes have any scenarios. If the **-corners** option is given, the command is applied to all scenarios of the specified corners and the current mode, and it is an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are given, the command is applied to all scenarios of the specified corners and the specified modes, and it is an error if there are no such scenarios. It is an error to give **-scenarios** with **-modes** or **-corners**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes uncertainty information from a clock named *CLK* and a pin named *clk_buf/Z*.

```
prompt> remove_clock_uncertainty [get_clocks CLK]
prompt> remove_clock_uncertainty [get_pins clk_buf/Z]
```

The following example removes interclock uncertainties between the *PHI1* and *PHI2* clock domains.

```
prompt> remove_clock_uncertainty -from PHI1 -to PHI1
prompt> remove_clock_uncertainty -from PHI2 -to PHI2
prompt> remove_clock_uncertainty -from PHI1 -to PHI2
prompt> remove_clock_uncertainty -from PHI2 -to PHI1
```

SEE ALSO

all_clocks(2)
create_clock(2)
report_clocks(2)
set_clock_latency(2)
set_clock_transition(2)
set_clock_uncertainty(2)

remove_clocks

Removes one or more clocks from the current design.

SYNTAX

string **remove_clocks** -all | *clock_list*

list *clock_list*

ARGUMENTS

-all

Specifies to remove all clocks in the current design.

clock_list

Specifies a list of collections containing clocks or patterns matching the clock names.

DESCRIPTION

Removes the specified clock objects from the current design. You must specify either *clock_list* or **-all**. The command returns the number of clocks removed.

If a removed clock is the only member of a path group, the path group is also removed. For information about path groups, refer to the **group_path** man page. To list all clock sources in the design, use **report_clock**.

All arrival and required times specified by commands **set_input_delay** and **set_output_delay** relatively to the clock that is being removed are also removed.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example removes clock CLK1.

prompt> **remove_clocks CLK1**

The following example removes all clocks from the current design.

prompt> **remove_clocks -all**

SEE ALSO

create_clock(2)
current_design(2)
get_clocks(2)
group_path(2)
report_clock(2)
set_input_delay(2)
set_output_delay(2)
reset_design(2)

remove_colors

Clears user specified colors from given objects and associated cells.

SYNTAX

status **remove_colors**
[*object_list*]

Data Types

object_list collection

ARGUMENTS

object_list

For cells in the object_list collection, user specified colors of all cells in the collection, and all cells descended from hierarchical cells in the collection will be removed.

For voltage areas in the object_list collection, user specified colors of all voltage areas in the collection, and all cells associated with the voltage areas will be removed.

If omitted, colors are removed from all cells in the current top design.

DESCRIPTION

If *objects* is specified and contains cells, this command removes user specified colors from the cells in the collection and all cells descended from hierarchical cells in the collection. If *objects* is specified and contains designs, this command removes user specified colors from all cells of designs in the collection. If *objects* contains voltage areas, this command removes user specified colors from the voltage areas in the collection and all cells associated with them. If *objects* is omitted or contains no cells, designs or voltage areas, the commands removes user specified colors on all objects in the current design whose color can be set by the set_colors command.

SEE ALSO

set_colors(2)

remove_command_hook

Remove hook from command execution

SYNTAX

string **remove_command_hook** -before *name*
-after *name*
-replace *name*
commandName

string *name*
string *commandName*

ARGUMENTS

-before *name*

Remove the before hook named *name*

-after *name*

Remove the after hook named *name*

-replace *name*

Remove the replace hook named *name*

commandName

Command to update

DESCRIPTION

The **remove_command_hook** removes a hook previously added to a command. When a hook is created via the **add_command_hook** the name of the hook is returned. Use that name to remove the hook. The get_command_hooks command can also be used to determine the name of each registered hook.

EXAMPLES

```
prompt> remove_command_hook place_opt -before before0
```

SEE ALSO

add_command_hook(2)
get_current_hook_command(2)
get_command_hooks(2)

remove_constraint_groups

Removes constraint groups from the current design.

SYNTAX

int **remove_constraint_groups**
-all | *constraint_group_list*

Data Types

constraint_group_list string or collection

ARGUMENTS

-all

Removes all constraint groups in the current design. This is mutually exclusive with *constraint_group_list*.

constraint_group_list

Specifies a list of constraint groups to remove. The list may contain constraint group collections, specified using the **get_constraint_groups** command. This is mutually exclusive with *-all*.

DESCRIPTION

This command removes all specified constraint groups. Constituent objects are disassociated from the removed constraint groups.

The number of removed constraint groups is returned.

EXAMPLES

The following example removes constraint groups:

```
prompt> set a [get_constraint_groups shield*]
{"shielding_0", "shielding_1"}

prompt> remove_constraint_groups $a
2
```

The following example removes all constraint groups:

```
prompt> remove_constraint_groups -all
10
```

SEE ALSO

create_wire_matching(2)
create_length_limit(2)
create_differential_group(2)
create_net_shielding(2)
create_net_priority(2)
create_bus_routing_style(2)
get_constraint_groups(2)

remove_corners

Removes one or more corners in multi corner analysis.

SYNTAX

status **remove_corners**
-all | *corner_list*

Data Types

corner_list collection

ARGUMENTS

-all

Removes all corners from memory.

corner_list

Specifies a list of unique corner strings to remove from memory.

DESCRIPTION

This command removes the specified corners from memory. To list the corners in the design, use the **all_corners** command.

The command returns the number of corners that were removed.

EXAMPLES

The following example removes the two corners named corner1 and corner2.

```
prompt> remove_corners {corner1 corner2}
1
```

SEE ALSO

all_corners(2)
create_corner(2)
current_corner(2)
get_corners(2)
report_corner(2)
set_scenario_status(2)

remove_custom_shields

Removes shielding for the specified nets if the shielding was created with Custom Router previously.

SYNTAX

status **remove_custom_shields**
[-nets nets]
[-keep_session (false|true)]

Data Types

nets a collection of nets to process

ARGUMENTS

-nets *nets*

Specifies the collection of nets for which shielding witll be deleted. If not specified, all top level nets in the design will be processed. Nets without shielding shapes created by the custom router will be skipped.

-keep_session *bool*

Specifies whether or not the custom router view of the data will be retained after the command completes. Use true only if you expect to perform additional custom route commands immediately after the current command. The default is false.

DESCRIPTION

The **remove_custom_shields** command removes shielding created by the custom router for a set of specified nets.

remove_cut_metals

This command removes cut metals at metal cut ends, where the cut metals are generated by **create_cut_metals** command.

SYNTAX

status **remove_cut_metals**

Data Types

DESCRIPTION

This command deletes cut metal shapes of top-design, where the cut metals shapes are created by **create_cut_metals** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

prompt> **remove_cut_metals**

SEE ALSO

remove_cut_metals(2)

remove_data_check

Removes specified data-to-data checks previously set by the **set_data_check** command.

SYNTAX

string **remove_data_check**
  -modes *mode_list*
  -corners *corner_list*
  -scenarios *scenario_list*
  -from *from_object*
    | -rise_from *from_object*
    | -fall_from *from_object*
  -to *to_object*
    | -rise_to *to_object*
    | -fall_to *to_object*
  [-setup | -hold]
  [-clock *clock*]
  [-all]

Data Types

from_object collection
to_object collection
clock_object collection

ARGUMENTS

-modes *mode_list*

Removes data check for scenarios of these modes (default is current mode, or current scenario if neither mode nor corner is specified)

-corners *corner_list*

Removes data check for scenarios of these corners (default is current scenario if neither mode nor corner is specified)

-scenarios *scenario_list*

Removes data check for these scenarios (default is current scenario)

-from *from_object*

Specifies a pin or port in the current design as the related pin of the data-to-data check to be removed. Both rising and falling checks are removed. You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-to *to_object*

Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be created. Both rising and falling checks are removed. You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-rise_from *from_object*

Specifies a pin or port in the current design as the related pin of the data-to-data check to be removed. This is similar to the **-from** option, but applies to only rising delays at the related pin. You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-fall_from *from_object*

Specifies a pin or port in the current design as the related pin of the data-to-data check to be removed. This is similar to the **-from** option, but applies to only falling delays at the related pin. You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-rise_to *to_object*

Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be created. This is similar to the **-to** option, but applies to only rising delays at the constrained pin. You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-fall_to *to_object*

Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be created. This is similar to the **-to** option, but applies to only falling delays at the constrained pin. You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-setup

Indicates that only the setup data check is to be removed. If neither **-setup** nor **-hold** is specified, both setup and hold checks are removed.

-hold

Indicates that only the hold data check is to be removed. If neither **-setup** nor **-hold** is specified, both setup and hold checks are removed.

-clock *clock_object*

Indicates that the data check for the specified clock at the related pin is to be removed. This option applies only if a previous **set_data_check** command used the **-clock** option to specify the same clock; otherwise, this option is ignored.

-all

Indicates that all the data checks to be removed. The command ignores all other options and removes all data checks.

DESCRIPTION

The **remove_data_check** command specifies data-to-data check(s) to be removed between the *from* object and the *to* object for the specified options. These checks were previously set using the **set_data_check** command.

Multicorner-Multimode Support

This command applies to the current scenario only.

EXAMPLES

The following example removes a data-to-data check from and1/B to and1/A with respect to the rising edge of signals at and1/B. Both setup and hold checks are removed.

```
prompt> remove_data_check -rise_from and1/B -to and1/A
```

The following example removes setup data-to-data checks between and1/B and and1/A with respect to the rising edge of signals at and1/B and the falling edge of signals at and1/A, coming from startpoints triggered by the CK1 clock:

```
prompt> remove_data_check -rise_from and1/B \
  -fall_to and1/A -setup -clock [get_clock CK1]
```

SEE ALSO

report_timing(2)
set_data_check(2)
update_timing(2)
timing_enable_multiple_clocks_per_reg(3)

remove_density_rules

Removes density rules from the specified technology data.

SYNTAX

status **remove_density_rules**
[-tech *tech*]
[-library *library*]
[-all]
[*density_rules*]

Data Types

tech collection
library collection
density_rules collection

ARGUMENTS

-tech *tech*

Specifies the technology data from which to remove the density rules.

If you do not specify the **-tech** or **-library** option, the command removes the rules from the technology data of the current library.

-library *library*

Specifies the library from which to remove the density rules.

If you do not specify the **-tech** or **-library** option, the command removes the rules from the technology data of the current library.

-all

Removes all density rules from the specified technology data.

The **-all** option and **density_rules** argument are mutually exclusive; you must specify one of these arguments.

density_rules

Specifies the density rules to remove. You can specify any combination of rule names, patterns, or collections, which are created by the **get_density_rules** command.

The **-all** option and **density_rules** argument are mutually exclusive; you must specify one of these arguments.

DESCRIPTION

The **remove_density_rules** command removes the specified density rules from the specified technology data.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

You can use either of the commands in the following example to remove all density rules from the technology data in the current library.

```
prompt> remove_density_rules -all
```

```
prompt> remove_density_rules [get_density_rules *]
```

The following example removes the density rules associated with the M3 layer from the technology data in the current library.

```
prompt> remove_density_rules \
  [get_density_rules -filter "layer_name==M3"]
1
```

SEE ALSO

create_density_rule(2)
get_density_rules(2)

remove_design_rules

Removes design rules (design_rule) from the current technology object.

SYNTAX

integer **remove_design_rules**
[-all]
[*design_rule_list*]

Data Types

design_rule_list collection

ARGUMENTS

-all

Removes all design rules.

The **-all** option and *design_rule_list* argument are mutually exclusive; you must specify one.

design_rule_list

Specifies the design rules to remove. You can specify the rules by name or as a collection.

The **-all** option and *design_rule_list* argument are mutually exclusive; you must specify one.

DESCRIPTION

This command removes design rules from the current technology object.

This command returns the number of rules removed. If there is a command syntax error, it raises a TCL_ERROR.

EXAMPLES

The following example removes the design rule named DESIGN_RULE_0.

prompt> **remove_design_rules DESIGN_RULE_0**

1

SEE ALSO

create_design_rule(2)
get_design_rules(2)
report_design_rules(2)
set_design_rule_attribute(2)

remove_dff_trace_filters

Removes filter patterns from the list of filters used by the **compute_dff_connections** command

SYNTAX

compute_dff_connections
[-blocks *list_of_blocks*]
-type pin | net
-patterns *list_of_patterns* | -all

Data Types

list_of_blocks list
list_of_patterns list

ARGUMENTS

-blocks *list of blocks*

Specifies a list of block names from which to remove the specified filter patterns. If not specified, the patterns are removed from the special "all blocks" list.

-type pin | net

Specifies the type of filter to remove, either *pin* or *net*.

-patterns *list_of_patterns*

Specifies a list of patterns to remove from the specified filter type list. Patterns must match existing patterns exactly (no wildcard support during removal).

-all

Removes all patterns from the specifies blocks, or if **-blocks** is not specified, from all block lists including the special "all blocks" list.

DESCRIPTION

This command removes trace filter patterns used by the Data Flow Flylines (DFF) tracer. See the **compute_dff_connections** command for more information.

A GUI interface to manage filter patterns is available as part of the Data Flow Flylines tool in the GUI. You can access the interface

with the Reload option.

GENERAL USAGE

This command removes one or more patterns from a list of existing patterns used by the **compute_dff_connections** command when tracing through the netlist.

EXAMPLES

The following example removes a pattern "M1/*" from the trace filter list previously added for block B1.

```
prompt> remove_dff_trace_filters -type pin -blocks B1 -patterns "M1/*"
1
```

SEE ALSO

compute_dff_connections(2)
create_dff_trace_filters(2)
write_dff_trace_filters(2)

remove_dft_location

Removes DFT hierarchy location specifications for the current design.

SYNTAX

status **remove_dft_location**
[-type *logic_type*]

Data Types

logic_type string

ARGUMENTS

-type *logic_type*

Specifies the types of DFT hierarchy location specifications to be removed from the current design. The following values are valid for *logic_type*:

CODEC
WRAPPER
PLL
SERIAL_CNTRL

You can specify a list of multiple types with this option.

To remove a default DFT hierarchy location specification that was applied using the **set_dft_location** command without the **-include** option, do not specify a value for this option.

DESCRIPTION

The **remove_dft_location** command removes DFT hierarchy location specifications for the current design.

EXAMPLES

The following example removes the DFT hierarchy location for the design named *top*:

prompt> **current_design top**

```
prompt> set_dft_location -include {CODEC} core_inst/CODEC

prompt> remove_dft_location -type {CODEC}
Removed DFT location 'core_inst/CODEC' for type 'CODEC' for design 'top'.
1
```

SEE ALSO

report_dft(2)
set_dft_location(2)

remove_dft_signal

Removes from specified ports the attributes that identify those ports as DFT signals in the current design.

SYNTAX

status **remove_dft_signal**
[-port *port_list*]
[-test_mode *mode_name_list*]
[-hookup_pin *pin_name*]

Data Types

port_list list
mode_name_list list
pin_name string

ARGUMENTS

-port *port_list*

Specifies a list of ports to which the command applies.

-test_mode *mode_name_list*

Specifies the mode(s) to which the command applies. The default mode is **Internal_scan**. Specifying **all** as the *mode_list* indicates that the specification applies to all modes. Any mode other than the default mode or **all** must be created before running the command with this option.

-hookup_pin *pin_name*

Specifies the hookup pin to which the command applies.

DESCRIPTION

The **remove_dft_signal** command removes from specified ports the attributes that identify those ports as DFT signal ports in the current design. These attributes were placed on the ports with the **set_dft_signal** command.

EXAMPLES

The following example removes the DFT signal attributes from the *TM* port:

```
prompt> remove_dft_signal -port TM
```

SEE ALSO

compile(2)
insert_dft(2)
preview_dft(2)
report_dft_signal(2)
report_dft(2)
set_dft_signal(2)

remove_disable_clock_gating_check

Restores clock gating checks previously disabled by the **set_disable_clock_gating_check** command.

SYNTAX

string **remove_disable_clock_gating_check**
object_list

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of cells and pins for which the clock gating check is restored.

DESCRIPTION

The **remove_disable_clock_gating_check** command restores clock gating checks previously disabled by the **set_disable_clock_gating_check** command. These two commands only affect automatically inferred clock gating checks, not library-defined checks.

Specify the list of cells and pins for which to restore the clock gating checks. The command restores clock gating checks for those objects.

The command does not tell you whether any clock gating checks are being performed on the specified objects or whether any such checks were previously disabled. To get this type of information, use the **report_clock_gating_check** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following command restores clock gating checks on cell U123 previously disabled by the **set_disable_clock_gating_check** command.

prompt> **remove_disable_clock_gating_check [get_cells U123]**

The following command restores clock gating checks that use pin U123/I as a gating enable pin or gating clock pin, previously disabled by the **set_disable_clock_gating_check** command.

prompt> **remove_disable_clock_gating_check [get_pins U123/I]**

SEE ALSO

report_clock_gating_check(2)
set_disable_clock_gating_check(2)

remove_disable_timing

Enables the previously disabled timing arcs.

SYNTAX

string **remove_disable_timing**
  [-from *from_pin_name*]
  [-to *to_pin_name*]
  [-loop_break]
  object_list

string *from_pin_name*
string *to_pin_name*
list *object_list*

ARGUMENTS

-from *from_pin_name*

-to *to_pin_name*

Specifies that only the arcs between these two pins on the specified cell or library cell be disabled.

-loop_break

Indicate whether constraints coming from tool, not user.

object_list

Specifies a list of cells, pins, library cells, library cell pins, or ports. The *object_list* must contain only cells or library cells (if **-from** and **-to** are specified).

DESCRIPTION

Restore timing arcs previously disabled with the **set_disable_timing** command. When **-from** and **-to** pins are specified, all arcs between these two pins on the cell or library cell are restored.

To list the timing arcs for a library cell, use **report_lib -timing_arcs**.

Multicorner-Multimode Support

EXAMPLES

The following example restores disabled setup and hold arcs between pins CLK and TE on library cell SFF1.

prompt> **remove_disable_timing -from CLK -to TE tech_lib/SFF1**

The following example restores all disabled cell arcs from or to pin A on cell U1/U2.

prompt> **remove_disable_timing U1/U2/A**

SEE ALSO

report_lib(2)
set_disable_timing(2)
report_lib(2)

remove_drc_error_data

Removes DRC error data objects.

SYNTAX

status **remove_drc_error_data**
drc_error_data
[-force]

Data Types

drc_error_data collection

ARGUMENTS

drc_error_data

Specifies the collection of DRC error data objects to remove.

-force

Removes the specified DRC error data object, regardless of pending multiple opens that have not been matched with a close.

DESCRIPTION

The **remove_drc_error_data** command removes the specified DRC error data objects. The command first closes the error data objects, which decrements the open count for the error data objects. If the open count decrements to zero, the command removes the error data object from memory. If the error data object was created as an external error data file, the error data file is removed from disk.

The open count for a DRC error data object is incremented each time you run the **create_drc_error_data** or **open_drc_error_data** command on it. The open count is decremented each time you run the **close_drc_error_data** command. The error data object is removed from memory when the open count reaches zero.

Use the **-force** option to remove the error data object from memory and remove the external error data file from disk, regardless of the open count value.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the DRC error data object named "dppinassgn.err".

```
prompt> remove_drc_error_data dppinassgn.err
1
```

The following example first opens an error data object named "dppinassgn.err", and then deletes it.

```
prompt> set data [open_drc_error_data -all dppinassgn.err]
{"dppinassgn.err"}
prompt> remove_drc_error_data $data
1
```

SEE ALSO

close_drc_error_data(2)
create_drc_error_data(2)
get_drc_error_data(2)
open_drc_error_data(2)
save_drc_error_data(2)

remove_drc_error_types

Removes physical DRC error type objects.

SYNTAX

status **remove_drc_error_types**
-error_data *drc_error_data*
drc_error_types

Data Types

drc_error_data collection
drc_error_types collection

ARGUMENTS

-error_data *drc_error_data*

Specifies the error data from which to remove physical DRC error types.

drc_error_types

A collection of physical DRC error type objects to remove.

DESCRIPTION

The **remove_drc_error_types** command removes physical DRC error type objects from the given error data. When an error type is removed, all errors of the type are also deleted from the error data.

EXAMPLES

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then removes all error_types that have the error_class value of **short**.

```
prompt> set data [open_drc_error_data -file_name my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
prompt> remove_drc_error_types -error_data $data \
  [get_drc_error_types -error_data $data -filter {error_class==short}]
```

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then removes the error type named "pin short".

```
prompt> set data [open_drc_error_data -file_name my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
prompt> remove_drc_error_types -error_data $data \
  [get_drc_error_types -error_data $data {"pin short"}]
```

SEE ALSO

create_drc_error_type(2)
get_drc_error_types(2)
remove_drc_errors(2)
remove_drc_error_data(2)
open_drc_error_data(2)

remove_drc_errors

Removes physical DRC error data objects.

SYNTAX

status **remove_drc_errors**
-error_data *drc_error_data*
drc_errors

Data Types

drc_error_data collection
drc_errors collection

ARGUMENTS

-error_data *drc_error_data*

Specifies the error data from which to remove errors.

drc_errors

A collection of physical DRC error data objects to remove.

DESCRIPTION

The **remove_drc_errors** command removes physical DRC error objects from the given error data.

EXAMPLES

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then removes all errors that have the status value of **fixed**.

```
prompt> set data [open_drc_error_data -file_name my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
prompt> remove_drc_errors -error_data $data \
    [get_drc_errors -error_data $data -filter {status==fixed}]
```

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then removes all errors that are of type "pin short".

```
prompt> set data [open_drc_error_data -file_name my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
prompt> set pinShortType [get_drc_error_types -error_data $data {"pin short"}]
{"pin short"}
prompt> remove_drc_errors -error_data $data \
    [get_drc_errors -error_data $data -of_objects $pinShortType]
```

SEE ALSO

create_drc_error(2)
open_drc_error_data(2)
remove_drc_error_data(2)
remove_drc_error_types(2)

remove_drive_resistance

Removes drive resistance for input or inout ports.

SYNTAX

string **remove_drive_resistance**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
port_list

Data Types

mode_list list
corner_list list
scenario_list list
port_list list

ARGUMENTS

-modes *mode_list*

Specifies the scenarios from which to remove the drive resistance. If this option is given, all scenarios of the specified modes will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners *corner_list*

Specifies the scenarios from which to remove the drive resistance. If this option is given, all scenarios of the specified corners and the current mode will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios *scenario_list*

Specifies the scenarios from which to remove the drive resistance. The **-modes** or **-corners** option must not be specified with this option.

port_list

Specifies a list of input or inout port names in the current design from which to remove the drive values.

DESCRIPTION

The **remove_drive_resistance** command removes the drive resistance value from one or more input or inout ports. This is equivalent to using **set_drive 0**. In fact, that is how this command works. If output ports are passed into **remove_drive_resistance**, a DES-003 message is issued, indicating that **set_drive** could not be performed on those ports.

Port drive resistance is managed on a per-scenario basis. For designs with multiple scenarios, you can specify different drive resistance for different scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command applies to the current scenario. The tool issues an error if there is no current scenario. If the **-scenarios** option is given, the command applies to all of the specified scenarios.

If the **-modes** option is given, the command applies to all scenarios of the specified modes. The tool issues an error if none of the specified modes have any scenarios.

If the **-corners** option is given, the command applies to all scenarios of the specified corners and the current mode. The tool issues an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are given, the command applies to all scenarios of the specified corners and the specified modes. The tool issues an error if there are no such scenarios. It is an error to specify **-scenarios** with **-modes** or **-corners**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example sets a drive resistance of 5 on ports A, B, and C, reports the new drive resistance, removes the drive resistance on port A, and reports the resistance after removal. In the final report, only drive resistance for ports B and C is reported.

```
prompt> set_drive_resistance 5 [get_ports {A B C}]
1

prompt> report_ports -drive {A B C}

****************************************
Report : port
Design : counter
****************************************


            Resistance        Transition
Input Port  Rise     Fall     Rise     Fall
------------------------------------------------------------
A           5.00     5.00     --       --
B           5.00     5.00     --       --
C           5.00     5.00     --       --
1

prompt> remove_drive_resistance A
1

prompt> report_ports -drive {A B C}

****************************************
Report : port
```

```
Design : counter
****************************************

                Resistance          Transition
Input Port    Rise      Fall      Rise      Fall
--------------------------------------------------------------------
A              --        --        --        --
B            5.00      5.00        --        --
C            5.00      5.00        --        --

1
```

SEE ALSO

report_ports(2)
set_load(2)
set_drive(2)

remove_driving_cell

Removes port driving cell information.

SYNTAX

string **remove_driving_cell**
[-rise]
[-fall]
[-min]
[-max]
[-clock *clock_name*]
[-clock_fall]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
port_list

Data Types

clock_name string
mode_list list
corner_list list
scenario_list list
port_list list

ARGUMENTS

-rise

Removes driving cell information only for the rising port transition.

-fall

Removes driving cell information only for the falling port transition.

-min

Removes driving cell information only for minimum timing analysis.

-max

Removes driving cell information only for maximum timing analysis.

-clock *clock_name*

Removes the driving cell set relative to the specified clock.

-clock_fall

Removes the driving cell relative to the falling edge of the clock. The default is the rising edge.

-modes ***mode_list***

Specifies the scenarios from which to remove the driving cells. If this option is given, all scenarios of the specified modes will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners ***corner_list***

Specifies the scenarios from which to remove the driving cells. If this option is given, all scenarios of the specified corners and the current mode will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios ***scenario_list***

Specifies the scenarios from which to remove the driving cells. The **-modes** or **-corners** option must not be specified with this option.

port_list

Provides a list of input or output ports.

DESCRIPTION

Removes driving cell information from the specified ports. The driving cell information is specified with the **set_driving_cell** command. The default is to remove all **-rise** and **-fall** information.

To see port drive information, use the **report_ports -drive** command.

Driving cells are managed on a per-scenario basis. For designs with multiple scenarios, you can specify scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command applies to the current scenario. The tool issues an error if there is no current scenario. If the **-scenarios** option is given, the command applies to all of the specified scenarios.

If the **-modes** option is given, the command applies to all scenarios of the specified modes. The tool issues an error if none of the specified modes have any scenarios.

If the **-corners** option is given, the command applies to all scenarios of the specified corners and the current mode. The tool issues an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are given, the command applies to all scenarios of the specified corners and the specified modes. The tool issues an error if there are no such scenarios. It is an error to specify **-scenarios** with **-modes** or **-corners**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes all driving cell information for port IN2.

```
prompt> remove_driving_cell IN2
```

SEE ALSO

report_ports(2)
set_driving_cell(2)

remove_duplicate_timing_contexts

Removes duplicate Scenarios, Modes and Corners for the current block

SYNTAX

int **remove_duplicate_timing_contexts**
[-analyze_only]
[-exclude_group_path_directives]

ARGUMENTS

-analyze_only

Only perform minimization analysis.

-exclude_group_path_directives

Disregard the group_path directive, which may allow further mode minimization.

DESCRIPTION

This command removes duplicate Scenarios, Modes and Corners. Scenarios that are determined to be duplicates and have the same Scenario status values are removed. Scenarios of Modes that are determined to be duplicate are re-assigned to a common Mode. Corners may be added to achieve that reassignment. Additional Corners will be named consistently to the Corner they were created from, with an _d suffix, where d is an integer. Scenarios of Corners that are determined to be duplicate are reassigned to a common Corner.

If the reduction of Modes, Corners or Scenarios is possible, it is expected that both runtime and capacity will be improved.

Switching activity is not considered when determining duplicate Modes. It may therefore be necessary to re-annotate switching activity manually where mode reduction was achieved.

If the design makes use of the **set_block_to_top_map** command to map Block to top corners, this mapping will need to be manually updated when a mapped top corner is removed. The **report_block_to_top_map** command may be used to determine the current mapping.

This command writes temporary files into the directory specified by the **shell.common.tmp_dir_path** application option. If the command reports an error in creating files, check the value of the application option and ensure that the directory is writable.

Multicorner-Multimode Support

This command uses information from all scenarios

EXAMPLES

The following example uses the **remove_duplicate_timing_contexts** command to analyze the timing contexts of the design and implement all possible mode, corner and scenario reductions. Any reduction is expected to reduce the design's memory footprint and runtime through the subsequent design flow. In order to reduce the current memory footprint it is necessary to save the design, quit the tool session and restart the tool session.

```
prompt> save_block -as des/pre-reduction;
prompt> report_qor -summary -include {setup hold};
prompt> remove_duplicate_timing_contexts;
...
Minimization Summary:
Modes    : Reduced from 4 to 2
Corners  : No reduction was possible
Scenarios: Reduced from 10 to 8
1
prompt> report_qor -summary -include {setup hold};
prompt> save_block -as des/post-reduction;
prompt> exit;
1
```

SEE ALSO

create_corner(2)
create_mode(2)
create_scenario(2)
report_block_to_top_map(2)
report_qor(2)
set_block_to_top_map(2)
set_scenario_status(2)
set_switching_activity(2)
shell.common.tmp_dir_path(3)

remove_eco_bus_buffer_patterns

Removes the specified ECO bus buffer pattern.

SYNTAX

status **remove_eco_bus_buffer_patterns**
[*patterns* | -all]
[-verbose]

Data Types

patterns collection

ARGUMENTS

patterns

Specifies the list of ECO bus buffer patterns to remove.

-all

Removes all ECO bus buffer patterns.

-verbose

Reports the names of the ECO bus buffer patterns, in addition to the number of ECO bus buffer pattern, that were removed.

By default, the command reports only the number of ECO bus buffer patterns that were removed.

DESCRIPTION

Removes the specified ECO bus buffer patterns.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all ECO bus buffer patterns with names that start with "or2t".

```
prompt> remove_eco_bus_buffer_patterns or2t*
2
```

SEE ALSO

create_eco_bus_buffer_pattern(2)
get_eco_bus_buffer_patterns(2)
report_eco_bus_buffer_patterns(2)

remove_eco_repeater

Remove buffer cells.

SYNTAX

remove_eco_repeater
cell_list
[-keep_net_name *keep_net_name*]
[-lib_cell_input *input_pin*]
[-lib_cell_output *output_pin*]
[-verbose]

ARGUMENTS

cell_list

Specifies the repeaters to remove.

-keep_net_name *keep_net_name*

Specifies the names of the net to keep. It must be connected with either input or output pin of the specified repeater to remove. This is optional.

-lib_cell_input *input_pin*

Specifies the name of the input pin to connect if the removed cell has multiple input pins. This is optional.

-lib_cell_output *output_pin*

Specifies the name of the output pin to connect if the removed cell has multiple output pins. This is optional.

-verbose

Show the verbose information. This is optional.

DESCRIPTION

This command removes repeaters. The input or output net of such repeater, which is specified by *keep_net_name*, will be kept. The other input or output net will be removed.

This command will specify the input pin for multiple inputs cells and output pin for multiple outputs cell with the option *lib_cell_input* or *lib_cell_output*. The kept net must connect with either the specified input pin or output pin.

This command support transparent interface optimization(TIO) flow, which can remove buffer cells in sub blocks of abstract view from top block.

This command supports editing a verilog module in the module aspect, through **edit_module**.

This command will check the editability of the block before removing buffer cells.

This command honors design.eco_freeze_silicon_mode. When this app option is true, buffers to remove will be marked as spare cell and renamed by adding _Spare post-fix, instead of being removed from database.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes repeater buf1. Its input net will be kept.

```
prompt> edit_module h1m {remove_eco_repeater -keep_net_name input_net_n1 buf1 }
```

The following example removes repeater buf1. Its output net will be kept.

```
prompt> edit_module h1m {remove_eco_repeater -keep_net_name output_net_n1 buf1 }
```

SEE ALSO

add_eco_repeater(2)
edit_module(2)

remove_edit_groups

Removes edit groups from the current design.

SYNTAX

int **remove_edit_groups**
[-force]
[-verbose]
[-all]
[*edit_group_list*]

Data Types

edit_group_list list

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Displays verbose deletion information.

-all

Removes all edit groups from the current design.

edit_group_list

Specifies a list of edit groups to remove. The list may contain edit group names, patterns, or collections. A collection may be specified by using the **get_edit_groups** command.

RETURN VALUE

The number of removed edit groups.

DESCRIPTION

This command removes edit groups from the current design. Note that objects that are contained by an edit group at the time of removal are not themselves removed from the design, rather they are removed from the edit group and are no longer contained within any edit group.

EXAMPLES

The following example removes edit groups named "edit_group1" and "edit_group2".

```
prompt> remove_edit_groups { edit_group1 edit_group2 }
2
```

The following example removes edit groups that match the pattern "edit_group*".

```
prompt> remove_edit_groups edit_group*
3
```

The following example removes all edit groups in the design.

```
prompt> remove_edit_groups -all
5
```

SEE ALSO

create_edit_group(2)
get_edit_groups(2)
add_to_edit_group(2)
remove_from_edit_group(2)
report_edit_groups(2)

remove_ems_rules

Removes user-defined EMS rules.

SYNTAX

remove_ems_rules
pattern | -all

Data Types

pattern pattern/collection capturing EMS user-defined rules

ARGUMENTS

pattern

This is a mandatory argument. It can be the name of a rule or a pattern capturing names of multiple rules or a collection of EMS rules. EMS rules by given name or matching given pattern or part of collection, should be already available in EMS memory; otherwise the command will fail with an error message. Option -all and argument "pattern" are mutually exclusive. If the EMS rules have messages instantiated for them then such rules are not removed.

-all

This is a mandatory option. This is mutually exclusive with argument "pattern". This option removes all user-defined rules in current EMS memory. Again if there are messages which have been created for the EMS rules then such rules are not removed.

DESCRIPTION

Command **remove_ems_rules** removes the user-defined rules if they do not have EMS messages created for them already.

EXAMPLES

prompt> **create_ems_database abc.ems**
prompt> **create_ems_rule -name "TEMP-001" -severity "Info" -message "Pin:%pin has capacitance:%cap"**
prompt> **create_ems_rule -name "TMP-002" -severity "Error" -message "Cell %cell is not placed"**
prompt> **remove_ems_rules "TEMP-001"**
prompt> **create_ems_message -rule "TMP-002" -parameters "cell:U1"**

```
prompt> remove_ems_rules "TMP-002"
Warning: Failed to remove EMS rules. (EMS-142)
```

SEE ALSO

create_ems_rule(2)
create_ems_message(2)
create_ems_database(2)

remove_feasibility_constraints

Removes constraints from timing paths that were added by feasibility constraint files.

SYNTAX

status **remove_feasibility_constraints**
-input_dir *dir_name*

Data Types

dir_name string

ARGUMENTS

-input_dir *dir_name*

Specifies the name of the directory **dir_name** that contain the feasibility constraint files.

DESCRIPTION

The *remove_feasibility_constraints* command removes the timing constraints added to the current design by the feasibility constraints files. It uses the files present in the specified directory to revert back the design before these files was applied to it.

A **input_dir** is required to be specified for the command to be able to remove the feasibility constraints applied to the current design.

EXAMPLE

The following example removes the timing constraints added by write_feasibility_constraints.

prompt> **remove_feasibility_constraints -input_dir Multicycle_Path_2**

SEE ALSO

write_feasibility_constraints(2)

remove_feedthroughs

Removes feedthrough ports and nets from blocks.

SYNTAX

string **remove_feedthroughs**
[-cells *cell_collection*]
[-nets *net_collection*]
[-include_original_feedthroughs]

Data Types

cell_collection collection
net_collection collection

ARGUMENTS

-cells cell_collection

Specifies the blocks from which to remove feedthrough ports and nets. By default, the command removes feedthroughs from all blocks.

-nets net_collection

Specifies the nets from which to remove feedthrough ports. By default, the command removes feedthroughs from all nets.

-include_original_feedthroughs

Removes the original feedthrough ports and new feedthrough ports that are created by the **place_pins** command. If you specify this option, the tool removes these feedthrough ports and the connections to these ports at the top level. By default, the command preserves the original ports and their top level connections.

DESCRIPTION

This command removes feedthrough ports and nets that exist within the design. By default, the command removes all added feedthrough nets and ports, and preserves all original nets and ports, for all blocks. To apply the command to a specific set of nets, use the **-nets** option to specify a collection of nets. To apply the command to certain blocks, use the **-cells** option to specify a collection of blocks.

The **-include_original_feedthroughs** option removes original and new feedthrough nets and ports An original port is a port on a block that is a feedthrough port in the original netlist. By default the command looks at multiple levels of physical hierarchy (MPH support) for feedthrough removal. The **set_editability** can be used to enable/disable removal on individual reference-blocks or

levels of physical hierarchy. If a net is connected to even a single instance that is disabled for design planning such a net will be skipped for feedthrough removal.

All the redundant feedthroughs in the whole design will always be removed, as long as the parent block of the redundant feedthrough is enabled for design planning.

EXAMPLES

The following example removes feedthrough nets and ports on all nets and all blocks, for both new and existing feedthroughs.

```
prompt> remove_feedthroughs -include_original_feedthroughs
```

SEE ALSO

place_pins(2)
report_feedthroughs(2)
set_editability(2)

remove_fill_cells

Removes the specified fill cells from the current block.

SYNTAX

status **remove_fill_cells**
[-remove_reference]
[-all]
[*fill_cell_list*]

Data Types

fill_cell_list list

ARGUMENTS

-remove_reference

Remove the reference internal designs of the specified fill cells, if unused after removing the fill cells.

fill_cell_list

Specifies the fill cells to remove.

The *fill_cell_list* argument and **-all** option are mutually exclusive; you must specify one.

-all

Removes all fill cells from the current block.

The **-all** option and *fill_cell_list* argument are mutually exclusive; you must specify one.

DESCRIPTION

Removes the specified fill cells from the current block.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the fill cells whose name includes "_1".

```
prompt> get_fill_cells *_1*
{FILL_INST_1 FILL_INST_10}

prompt> remove_fill_cells -remove_reference *_1*
2
```

SEE ALSO

get_fill_cells(2)

remove_floorplan_rules

Removes the floorplan rules in the design.

SYNTAX

status **remove_floorplan_rules**
[-object_types *type_list*]
[-lib_cells *lib_cells*]
[-all]
[*rule_list*]

Data Types

type_list list
lib_cells collection
rule_list list

ARGUMENTS

-object_types *type_list*

Specifies the list of object types for which some floorplan rules are defined. Floorplan rules that have specified object types in their **-from_object_types** or **-to_object_types** or **-object_types** will be removed. Valid values for this option are *block_boundary*, *core_area*, *hard_macro*, *placement_blockage*, *routing_blockage*, *shape*, *soft_macro* and *std_cell_area*. This option is mutually exclusive with the **-all** and *rule_list* options.

-lib_cells *lib_cells*

Specifies the collection of library cells for which some floorplan rules are defined. Floorplan rules that have specified library cells in their **-from_lib_cells** or **-to_lib_cells** or **-lib_cells** will be removed. This option is mutually exclusive with the **-all** and *rule_list* options.

-all

Removes all floorplan rules defined in the design. This option is mutually exclusive with the **-object_types**, **-lib_cells** and *rule_list* options.

rule_list

Specifies the list of floorplan rule names to remove. This option is mutually exclusive with the **-object_types**, **-lib_cells** and **-all** options.

DESCRIPTION

The **remove_floorplan_rules** command removes existing floorplan rules in a design based on specified input parameters. On successful removal, the command returns a status of 1.

The *rule_list* option is mutually exclusive with the **-object_types**, **-lib_cells** and **-all** options. If any named rule in *rule_list* does not exist or no rule exists for the given object types or library cells, a warning is issued.

EXAMPLES

The following example tries to remove two floorplan rules named e2 and ab. The floorplan rule ab does not exist.

```
prompt> remove_floorplan_rules {e2 ab}

Warning: Floorplan rule 'ab' does not exist.
0
prompt> report_floorplan_rules e2
Warning: Floorplan rule 'e2' does not exist.
0
```

The following example tries to remove floorplan rules for object type std_cell_area and library cell lc4. Since no floorplan rule exists for library cell lc4, the command issues a warning message and removes floorplan rule e2 for object type std_cell_area.

```
prompt> remove_floorplan_rules -object_types std_cell_area -lib_cells \
  [get_lib_cells */lc4]

Warning: No floorplan rules exists for given lib cells.
1
prompt> report_floorplan_rules e2
Warning: Floorplan rule 'e2' does not exist.
0
```

SEE ALSO

report_floorplan_rules(2)
set_floorplan_area_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_extension_rules(2)
set_floorplan_forbidden_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_length_rules(2)
set_floorplan_spacing_rules(2)
set_floorplan_width_rules(2)

remove_from_bound

Unassigns cells and ports from a bound in the current design.

SYNTAX

int **remove_from_bound**
bound_object
cell_and_port_list

Data Types

bound_object list
cell_and_port_list list

ARGUMENTS

bound_object

Specifies the name or collection of a single bound. The bound object can be specified by using the **get_bounds** command. However it must return a collection of one bound object.

cell_and_port_list

Specifies a list of cells and ports to unassign from the bound. The list may contain names, patterns, or collections. If a cell or port is not assigned to the bound, it is skipped and not removed. Cells that are implicitly in the bound (because its parent hierarchical cell is the bound) cannot be unassigned from the bound. Only assigned cells may be unassigned. However such cells may be assigned to a different bound using the **add_to_bound** command.

DESCRIPTION

The **remove_from_bound** command unassigns cells and ports from an existing bound, which are currently assigned to the bound.

You can readjust the utilization by unassigning cells from the bound. The placer assumes that the cells are no longer constrained by placement bounds.

EXAMPLES

The following example unassigns all cells from a bound named "movebound1".

```
prompt> remove_from_bound movebound1 *
1
```

The following example unassigns a hierarchical cell from a bound named "movebound2".

```
prompt> remove_from_bound movebound2 mid
1
```

SEE ALSO

create_bound(2)
get_bounds(2)
add_to_bound(2)
report_bounds(2)
remove_bounds(2)

remove_from_bundle

Removes one or more objects from the specified bundle.

SYNTAX

status_value **remove_from_bundle**
-bundle *bundle*
[-all]
object_list

Data Types

bundle collection
object_list list

ARGUMENTS

-bundle *bundle*

Specifies the bundle to remove object from.

-all

Removes all occurrences of the specified object from the bundle. By default, only the first occurrence of the object is removed.

object_list

Specifies the list of objects to remove from the bundle.

DESCRIPTION

Removes a list of objects from the specified bundle.

EXAMPLES

The following example removes objects from a bundle.

prompt> **remove_from_bundle -bundle [get_bundles BUNDLE_5] [get_nets Clk]**

1

SEE ALSO

add_to_bundle(2)
create_bundle(2)
get_bundles(2)
remove_bundles(2)
report_bundles(2)

remove_from_collection

Removes objects from a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

collection **remove_from_collection**
[-intersect]
collection1
object_spec

Data Types

collection1	collection
object_spec	list

ARGUMENTS

-intersect

Removes objects from collection1 not found in object_spec. Without this option, removes objects from collection1 that are found in object_spec.

collection1

Specifies the base collection to be copied to the result collection. Objects matching *object_spec* are removed from the result collection.

object_spec

Specifies a list of named objects or collections to remove. The object class of each element in this list must be the same as in the base collection. If the name matches an existing collection, the collection is used. Otherwise, the objects are searched for in the database using the object class of the base collection.

DESCRIPTION

The **remove_from_collection** command removes elements from a collection, creating a new collection.

If the base collection is homogeneous, any element of the *object_spec* that is not a collection is searched for in the database using the object class of the base collection. If the base collection is heterogeneous, any element of the *object_spec* that is not a collection is ignored.

If the *-intersect* option is not specified, which is the default mode, and if nothing matches the *object_spec*, the resulting collection is a

copy of the base collection. If everything in the *collection1* option matches the *object_spec*, the result is the empty collection. With the *-intersect* option the results are reversed.

For background on collections and querying of objects, see the **collections** man page.

EXAMPLES

The following example from PrimeTime gets all input ports except "CLOCK".

```
pt_shell> set cPorts [remove_from_collection [all_inputs] CLOCK]
{"in1", "in2"}
```

SEE ALSO

add_to_collection(2)
collections(2)

remove_from_edit_group

Removes objects from an edit group in the current design.

SYNTAX

int **remove_from_edit_group**
edit_group_object
object_list

Data Types

edit_group_object list
object_list list

ARGUMENTS

edit_group_object

Specifies the name or collection of a single edit group. The edit group object can be specified by using the **get_edit_groups** command, however, it must return a collection of exactly one edit group object.

object_list

Specifies a list of objects to remove from the edit group. The list can contain names, patterns, or collections. Objects which are not contained in the edit group are skipped.

DESCRIPTION

The **remove_from_edit_group** command removes objects from an existing edit group.

The command returns a collection containing the modified edit group as the only object, or an empty collection if the edit group was removed due to its ungroup_on_remove attribute setting. If the command fails, it return a Tcl error.

EXAMPLES

The following example removes all shapes from an edit group named "edit_group1".

```
prompt> remove_from_edit_group edit_group1 [get_shapes]
```

{"edit_group1"}

SEE ALSO

add_to_edit_group(2)
create_edit_group(2)
get_edit_groups(2)
remove_edit_groups(2)
report_edit_groups(2)

remove_from_group

Removes objects from the group in the current design.

SYNTAX

int **remove_from_group**
group
object_list

Data Types

group list
object_list list

ARGUMENTS

group

Specifies the name or collection of group. The group object can be specified by using the **get_groups** command.

object_list

Specifies a list of objects to remove from the group. The list can contain names, patterns, or collections. Removing an object which is not a member of the group is an error.

RETURN VALUE

The number of object removed the group or TCL_ERROR if the command fails.

DESCRIPTION

The **remove_from_group** command removes objects from the group.

EXAMPLES

The following example removes shape R1 from group named "group1".

```
prompt> remove_from_group group1 [get_shapes R1 ]
1
```

SEE ALSO

add_to_group(2)
create_group(2)
get_groups(2)
remove_groups(2)
report_groups(2)

remove_from_io_guide

Removes pad cells from an I/O guide in the current design.

SYNTAX

int **remove_from_io_guide**
io_guide_object
pad_cell_list

Data Types

io_guide_object list
pad_cell_list list

ARGUMENTS

io_guide_object

Specifies the name or collection of a single I/O guide. The *io_guide_object* can be specified with the **get_io_guides** command, however, it must return a collection of one I/O guide object.

pad_cell_list

Specifies a list of pad cells to remove from the I/O guide. The list can contain names, patterns, or collections. If a cell or port is assigned to another I/O guide, it is skipped and not removed.

DESCRIPTION

The **remove_from_io_guide** command removes pad cells from an existing I/O guide. The command returns 1 if successful, 0 if it fails, or a Tcl error if there is a command syntax error.

EXAMPLES

The following example removes all cells from a I/O guide named "io_guide1".

```
prompt> remove_from_io_guide io_guide1 *
1
```

SEE ALSO

add_to_io_guide(2)
create_io_guide(2)
get_io_guides(2)
remove_io_guides(2)
report_io_guides(2)

remove_from_io_ring

Removes I/O guides from an I/O ring in the current design.

SYNTAX

int **remove_from_io_ring**
io_ring_object
guide_list

Data Types

io_ring_object list
guide_list list

ARGUMENTS

io_ring_object

Specifies the name or collection of a single I/O ring. The *io_ring_object* can be specified by using the **get_io_rings** command. However it must return a collection of one I/O ring object.

guide_list

Specifies a list of I/O guides to remove from the I/O ring. The list can contain names, patterns, or collections. If a cell or port is assigned to another I/O ring, it is skipped and not removed.

DESCRIPTION

The **remove_from_io_ring** command removes I/O guides from an existing I/O ring. The command returns 1 if the successful, 0 if it fails, or a Tcl error if there is a command syntax error.

EXAMPLES

The following example removes all I/O guides from a I/O ring named "io_ring1".

```
prompt> remove_from_io_ring io_ring1 *
1
```

SEE ALSO

add_to_io_ring(2)
create_io_ring(2)
get_io_rings(2)
remove_io_rings(2)
report_io_rings(2)

remove_from_matching_type

Removes cells, pins, and terminals from the matching type.

SYNTAX

int **remove_from_matching_type**
[-auto_fix]
matching_type
object_list

Data Types

matching_type list
object_list list

ARGUMENTS

-auto_fix

Removes non-existent objects from the matching type. Objects become non-existent when reference libraries or designs are modified, and objects no longer exist in the cell hierarchy.

The **-auto_fix** and *object_list* options are mutually exclusive; you can specify only one of them.

matching_type

Specifies the name or collection of a single matching type. The matching type can be specified by using the **get_matching_types** command. However it must return a collection of one matching type.

object_list

Specifies a list of cells, pins, and terminals to remove from the matching type. The list may contain names, patterns, or collections. If an object is not in the matching type, it is skipped and not removed.

The **-auto_fix** and *object_list* options are mutually exclusive; you can specify only one of them.

DESCRIPTION

The **remove_from_matching_type** command removes cells, pins, and terminals from an existing matching type, which are currently in the matching_type.

EXAMPLES

The following example removes a cell from a matching type named "myMatchingType1".

```
prompt> remove_from_matching_type myMatchingType1 [get_cells inv1]
1
```

SEE ALSO

add_to_matching_type(2)
create_matching_type(2)
get_matching_types(2)
remove_matching_types(2)
report_matching_types(2)

remove_from_multisource_clock_sink_group

Removes sinks from a sink group defined for multisource clock tap assignment

SYNTAX

status **remove_from_multisource_clock_sink_group**
-name *name*
-sinks *pin_or_port_list*

Data Types

name	string
pin_or_port_list	collection

ARGUMENTS

-name *name*

This required option defines the name of the sink group from which sinks are to be removed. Sink groups are defined using the **create_multisource_clock_sink_group** command.

-sinks *pin_or_port_list*

This required option specifies a list of sinks that should get removed from the sink list of the sink group given by option **-name**.

DESCRIPTION

The **remove_from_multisource_clock_sink_group** command is used to remove clock tree sinks from a sink group defined for multisource clock tap assignment.

The list of assigned sinks to a sink group can be obtained by accessing the sink group attribute **sinks**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This example removes the clock tree sink reg1/CK from sink group sink_group1.

```
prompt> remove_from_multisource_clock_sink_group -name sink_group1 -sinks [get_pins reg1/CK]
```

SEE ALSO

add_to_multisource_clock_sink_group(2)
create_multisource_clock_sink_group(2)
get_multisource_clock_sink_groups(2)
remove_multisource_clock_sink_groups(2)
report_multisource_clock_sink_groups(2)
set_multisource_clock_tap_options(2)
synthesize_multisource_clock_taps(2)

remove_from_net

Removes shapes vias, and via_matrixes from a net in the current design.

SYNTAX

int **remove_from_net**
[-force]
net
shape_and_via_list

Data Types

net collection
shape_and_via_list collection

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

net

Specifies the net from which to remove shapes, vias and via_matrixes.

shape_and_via_list

Specifies the shapes, vias and via_matrixes to remove from the net. The list may contain names, patterns, or collections. If a shape, via or via_matrix is assigned to another net, it is skipped and not removed.

DESCRIPTION

The **remove_from_net** command removes shapes, vias and via_matrixes from an existing net.

EXAMPLES

The following example removes a shape from a net.

```
prompt> remove_from_net net1 [get_shapes RECT_18_10]
1
```

The following example removes a via from a net.

```
prompt> remove_from_net net1 [get_vias VIA_SA_132]
1
```

SEE ALSO

create_net(2)
get_nets(2)
add_to_net(2)
create_shape(2)
get_shapes(2)
create_via(2)
get_vias(2)
create_via_matrix(2)
get_via_matrixes(2)

remove_from_net_bus

Removes the specified net from the existing net bus in the current design.

SYNTAX

collection **remove_from_net_bus**
net_bus
net

Data Types

net_bus string or collection
net string or collection

ARGUMENTS

net_bus

Specifies the net bus from which to remove the net.

net

Specifies the net which needs to be removed from the net bus. The *net* should have the same base name as net bus.

RETURN VALUE

This command returns the updated net bus (as a collection), an empty string if it fails, or a TCL_ERROR if there is a command syntax error

DESCRIPTION

This command removes the specified net from an existing net bus in the current design. Individual nets will remain. Net removal is allowed only when no gaps are created in bus numbering and at least one net remains in the bus.

EXAMPLES

The following example creates a bus net named bus1 , then removes net bus1[7] from the net bus. The third command tries to remove a bit from bus net in the middle of the bus, which causes the tool to issue an error message.

```
prompt> create_net_bus -create_nets bus1[0:7]
{bus1}

prompt> remove_from_net_bus bus1 bus1[7]
{bus1}

prompt> remove_from_net_bus bus1 bus1[4]
Error: Net removal will create a hole in the net bus. (NDM-120)
```

SEE ALSO

add_to_net_bus(2)
create_net_bus(2)
get_net_buses(2)
remove_net_buses(2)
report_net_buses(2)

remove_from_pin_blockage

Removes pins, nets, and ports from pin blockages in the current design.

SYNTAX

int **remove_from_pin_blockage**
pin_blockage_list
object_list

Data Types

pin_blockage_list list
object_list list

ARGUMENTS

pin_blockage_list

Specifies the names or collection of pin blockages. The pin blockages can be specified by using the **get_pin_blockages** command.

object_list

Specifies a list of physical pins, nets, or ports to remove from the pin blockages.

DESCRIPTION

The **remove_from_pin_blockage** command removes pins, nets, and ports from existing pin blockages.

EXAMPLES

The following example removes all pins a pin blockage named "PIN_BLOCKAGE_1".

```
prompt> remove_from_pin_blockage PIN_BLOCKAGE_1 [get_pins *]
1
```

SEE ALSO

create_pin_blockage(2)
get_pin_blockages(2)
report_pin_blockages(2)
remove_pin_blockages(2)
add_to_pin_blockage(2)

remove_from_pin_guide

Removes pins, nets, and ports from pin guides in the current design.

SYNTAX

int **remove_from_pin_guide**
pin_guide_list
object_list

Data Types

pin_guide_list list
object_list list

ARGUMENTS

pin_guide_list

Specifies the names or collection of pin guides. The pin guides can be specified by using the **get_pin_guides** command.

object_list

Specifies a list of physical pins, nets, or ports to remove from the pin guides.

DESCRIPTION

The **remove_from_pin_guide** command removes pins, nets, and ports from existing pin guides.

EXAMPLES

The following example removes all pins a pin guide named "PIN_GUIDE_1".

```
prompt> remove_from_pin_guide PIN_GUIDE_1 [get_pins *]
1
```

SEE ALSO

create_pin_guide(2)
get_pin_guides(2)
report_pin_guides(2)
remove_pin_guides(2)
add_to_pin_guide(2)

remove_from_placement_attraction

Unassigns cells from a placement_attraction in the current design.

SYNTAX

int **remove_from_placement_attraction**
placement_attraction_object
cell_list

Data Types

placement_attraction_object list
cell_list list

ARGUMENTS

placement_attraction_object

Specifies the name or collection of a single placement_attraction. The placement_attraction object can be specified by using the **get_placement_attractions** command. However it must return a collection of one placement_attraction object.

cell_list

Specifies a list of cells to unassign from the placement_attraction. The list may contain names, patterns, or collections. If a cell is not assigned to the placement_attraction, it is skipped and not removed. Cells that are implicitly in the placement_attraction (because its parent hierarchical cell is the placement_attraction) cannot be unassigned from the placement_attraction. Only assigned cells may be unassigned.

DESCRIPTION

The **remove_from_placement_attraction** command unassigns cells from an existing placement_attraction, which are currently assigned to the placement_attraction.

EXAMPLES

The following example unassigns all cells from a placement_attraction named "placement_attraction1".

```
prompt> remove_from_placement_attraction placement_attraction1 *
1
```

The following example unassigns a hierarchical cell from a placement_attraction named "placement_attraction2".

```
prompt> remove_from_placement_attraction placement_attraction2 mid
1
```

SEE ALSO

create_placement_attraction(2)
get_placement_attractions(2)
add_to_placement_attraction(2)
report_placement_attractions(2)
remove_placement_attractions(2)

remove_from_port_bus

Removes the specified port from the existing port bus in the current design.

SYNTAX

collection **remove_from_port_bus**
port_bus
port

Data Types

port_bus string or collection
port string or collection

ARGUMENTS

port_bus

Specifies the port bus from which to remove the port.

port

Specifies the port which needs to be removed from the port bus. The *port* should have the same base name as port bus.

RETURN VALUE

This command returns the updated port bus (as a collection), an empty string if it fails, or a TCL_ERROR if there is a command syntax error

DESCRIPTION

This command removes the specified port from an existing port bus in the current design. Individual ports will remain. Port removal is allowed only when no gaps are created in bus numbering and at least one port remains in the bus.

EXAMPLES

The following example creates a bus port named bus1 , then removes port bus1[7] from the port bus. The third command tries to remove a bit from bus port in the middle of the bus, which causes the tool to issue an error message.

```
prompt> create_port_bus -create_ports bus1[0:7]
{bus1}

prompt> remove_from_port_bus bus1 bus1[7]
{bus1}

prompt> remove_from_port_bus bus1 bus1[4]
Error: Port removal will create a hole in the port bus. (NDM-115)
```

SEE ALSO

add_to_port_bus(2)
create_port_bus(2)
get_port_buses(2)
remove_port_buses(2)
report_port_buses(2)

remove_from_routing_corridor

Dissociates nets, supernets and their bundles from routing corridor.

SYNTAX

status **remove_from_routing_corridor**
routing_corridor
object_list

Data Types

routing_corridor string
object_list list

ARGUMENTS

routing_corridor

Specifies the name or collection of a single routing corridor. The routing corridor can be specified by using the **get_routing_corridors** command. However it must return a collection of only one routing corridor object.

object_list

Specifies the nets, supernets and their bundles to dissociate from their routing corridors. This is a required option.

DESCRIPTION

This command dissociates the specified nets, supernets and their bundles from the routing corridor.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example dissociates nets whose names start with "clock" from routing corridor RC_2.

prompt> **remove_from_routing_corridor RC_2 [get_nets clock*]**

```
1
```

The following example dissociates a bundle of supernets from routing corridor RC_2.

```
prompt> remove_from_routing_corridor RC_2 supernet_bundle_0
1
```

SEE ALSO

create_routing_corridor(2)
remove_routing_corridors(2)
report_routing_corridors(2)
add_to_routing_corridor(2)
remove_routing_corridor_shapes(2)
create_routing_corridor_shape(2)

remove_from_rp_group

Removes cell, relative placement group or relative placement blockage from a relative placement group.

SYNTAX

int **remove_from_rp_group**
rp_group_object
-cells *cell_list* | -rp_group *rp_group_object* | -blockage *blockage_object*

Data Types

rp_group_object collection
cell_list list or collection
blockage_object list or collection

ARGUMENTS

rp_group_object

Specifies the name or collection of a single relative placement group, from which the objects will be removed. The relative placement group can be specified using the **get_rp_groups** command. However it must return a collection of one and only one relative placement group.

-cells *cell_list*

Specifies a list of cells to remove from relative placement group. The list may contain names, patterns, or collections. It is an error to provide a cell, which does not belong to provided relative placement group.

-rp_group *hier_rp_group_object*

Specifies the relative placement group name or collection to remove from the relative placement group. It is an error to provide a relative placement group, which does not belong to provided relative placement group.

-blockage *blockage_object*

Specifies the relative placement blockage name or collection to remove from the relative placement group. It is an error to provide a relative placement blockage, which does not belong to provided relative placement group.

DESCRIPTION

The **remove_from_rp_group** command removes cells, relative placement group or relative placement blockage from existing

relative placement group. If the object being removed is at location specified for *rp_location* anchor corner then anchor corner will be removed from relative placement group.

EXAMPLES

The following example uses **remove_from_rp_group** to remove a cell from an existing relative placement group.

```
prompt> get_rp_groups grp_ripple
{grp_ripple}

prompt> remove_from_rp_group grp_ripple -cells carry_in_1
1
```

The following example uses **remove_from_rp_group** to remove a blockage from an existing relative placement group.

```
prompt> remove_from_rp_group grp_ripple -blockage grp_ripple_blkg2
1
```

SEE ALSO

create_rp_group(2)
get_rp_groups(2)
add_to_rp_group(2)
write_rp_groups(2)
remove_rp_groups(2)

remove_from_scan_chain

Removes the specified stub chains from the existing scan chain in the current design.

SYNTAX

collection **remove_from_scan_chain**
scan_chain
stub_chains

Data Types

scan_chain string or collection
stub_chains string or collection

ARGUMENTS

scan_chain

Specifies the scan_chain from which to remove stub chains.

stub_chains

Specifies the stub chains which needs to be removed from the scan chain. The *stub_chains* must be in the scan_chain.

DESCRIPTION

This command removes the specified stub_chains from an existing scan_chain in the current design. Individual stub chain will continue to exist.

EXAMPLES

The following example creates a scan chain named scan1 with a stub chain named stub1 added to it, then removes the stub chain stub1 from the scan chain.

```
prompt> create_stub_chain -name stub1
{bus1}

prompt> create_scan_chain -name scan1 {stub1}
```

```
{bus1}

prompt> remove_from_scan_chain scan1 {stub1}
1
```

SEE ALSO

add_to_scan_chain(2)
create_scan_chain(2)
get_scan_chains(2)
remove_scan_chains(2)
report_scan_chains(2)
create_stub_chain(2)

remove_generated_clock

Removes generated clock objects from the current design.

SYNTAX

status **remove_generated_clocks**
-all | *generated_clock_list*

Data Types

generated_clock_list list

ARGUMENTS

-all

Indicates that all generated clocks are to be removed.

generated_clock_list

Specifies a list of names of generated clocks to be removed.

DESCRIPTION

This command removes from the current design generated clocks previously created using the **create_generated_clock** command. The command returns the number of generated clocks removed. All attributes set on generated clocks (for example, **false_paths, multicycle_paths**) are also removed.

To display information about clocks and generated clocks in the design, use the **report_clocks** command.

Multicorner-Multimode Support

EXAMPLES

The following example removes the generated clock GEN1 from the design.

prompt> **remove_generated_clocks GEN1**

The following example removes all generated clocks from the design.

```
prompt> remove_generated_clocks -all
```

SEE ALSO

create_generated_clock(2)
report_clocks(2)

remove_generated_clocks

Removes generated clock objects from the current design.

SYNTAX

status **remove_generated_clocks**
-all | *generated_clock_list*

Data Types

generated_clock_list list

ARGUMENTS

-all

Indicates that all generated clocks are to be removed.

generated_clock_list

Specifies a list of names of generated clocks to be removed.

DESCRIPTION

This command removes from the current design generated clocks previously created using the **create_generated_clock** command. The command returns the number of generated clocks removed. All attributes set on generated clocks (for example, **false_paths, multicycle_paths**) are also removed.

To display information about clocks and generated clocks in the design, use the **report_clocks** command.

Multicorner-Multimode Support

EXAMPLES

The following example removes the generated clock GEN1 from the design.

prompt> **remove_generated_clocks GEN1**

The following example removes all generated clocks from the design.

```
prompt> remove_generated_clocks -all
```

SEE ALSO

create_generated_clock(2)
report_clocks(2)

remove_grids

Removes the specified grids from the current design.

SYNTAX

status **remove_grids**
-all | *grid*

Data Types

grid collection

ARGUMENTS

-all

Remove all grids from the current design. This option is mutually exclusive with the *grid* argument, you must specify **-all** or the *grid*.

grid

Specifies the grid object to be removed. You can specify a single object or a collection of grids. This option is mutually exclusive with the **-all** argument, you must specify **-all** or the *grid*.

DESCRIPTION

This command removes the specified grids from the current design. The grids could be of any type, block or user.

EXAMPLES

The following example removes all the grids in the current design:

prompt> **remove_grids -all**

The following example uses the *grid* argument to remove the grid named gr1:

prompt> **remove_grids [get_grids gr1]**

The following example uses the *grid* argument to remove all the user grids:

```
prompt> remove_grids [get_grids -type user]
```

SEE ALSO

create_grid(2)
get_grids(2)
report_grids(2)
set_block_grid_references(2)
set_grid(2)
snap_cells_to_block_grid(2)

remove_groups

Removes groups from the current design.

SYNTAX

int **remove_groups**
[-verbose]
[*group_list*]

Data Types

group_list list

ARGUMENTS

-verbose

Displays verbose deletion information.

group_list

Specifies a list of groups to remove. The list may contain group names, patterns, or collections. A collection may be specified by using the **get_groups** command. If *group_list* is not provided then it will delete all groups from the current design.

RETURN VALUE

The number of removed groups.

DESCRIPTION

This command removes groups from the current design. Note that objects that are contained by an group at the time of removal are not removed from the design, rather they are removed from that group.

EXAMPLES

The following example removes groups named "group1" and "group2".

```
prompt> remove_groups { group1 group2 }
2
```

The following example removes groups that match the pattern "group*".

```
prompt> remove_groups group*
3
```

The following example removes all groups in the design.

```
prompt> remove_groups
5
```

SEE ALSO

create_group(2)
get_groups(2)
add_to_group(2)
remove_from_group(2)
report_groups(2)

remove_host_options

Remove settings for multi-threaded and distributed processing created by **set_host_options**.

SYNTAX

status **remove_host_options**
[-all]
[-name *options_name*]
[-target ICC2 | ICV | PrimeTime | StarRC | ALL]

Data Types

options_name string

ARGUMENTS

-all

Specifies that all host options should be deleted.

-target ICC2 | **ICV** | **PrimeTime** | **StarRC** | **ALL**

The target tool of set of options to be deleted.

The user could also pass combination of -all, name and -target to selectively remove option settings.

DESCRIPTION

The **remove_host_options** command is used to unset the parameters set by the **set_host_options** command.

If the **-name** argument is given, any host options of that name will be deleted. If the **-all** argument is given, all host options will be deleted, and the **-max_cores** value for the main (parent) job will be restored to the default setting.

SEE ALSO

set_host_options(2)
report_host_options(2)

remove_hot_spots

Removes all data structure generated by the **generate_hot_spots** command in the previous run..

SYNTAX

status **remove_hot_spots**

ARGUMENTS

The **remove_hot_spots** command is used to clear all data structure created by the **generate_hot_spots** command. Within the same session, users are not allowed to invoke the **generate_hot_spots** command twice without first clearing any remaining data structure from the first **generate_hot_spots** command run.

SEE ALSO

generate_hot_spots(2)
report_hot_spots(2)

remove_ideal_latency

Removes ideal latency values from the specified objects.

SYNTAX

int **remove_ideal_latency**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-all]
[-rise] [-fall]
[-min] [-max]
object_list

list *object_list*

ARGUMENTS

-modes *mode_list*

Specifies the modes for which to remove the ideal latency value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is de-annotated separately. If this option is not given, the command removes latency for every scenario that is active with the specified corner(s). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to remove the ideal latency value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is de-annotated separately. If this option is not given, the command removes latency for every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios for which to remove the ideal latency value. Each of the specified scenarios is de-annotated separately. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command applies to the current scenario only.

-all

Removes the ideal latency time from every pin in the current design. If you specify **-all**, you cannot specify any other arguments.

-rise

Removes the rise ideal latency time. If you do not specify the **-rise** or **-fall** options, both values are removed.

-fall

Removes the fall ideal latency time. If you do not specify the **-rise** or **-fall** options, both values are removed.

-min

Removes the minimum ideal latency time. If you do not specify the **-min** or **-max** options, both values are removed.

-max

Removes the maximum ideal latency time. If you do not specify the **-min** or **-max** options, both values are removed.

object_list

Specifies a list of ports or pins from where to remove ideal latency.

DESCRIPTION

Removes the user-specified ideal latency that was previously set by the **set_ideal_latency** command from specified objects in *object_list*.

To list ideal latency values, use the **report_ideal_network** command.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes ideal latency from the output pin named *Z* of cell instance *U1/U2/U3*.

```
prompt> remove_ideal_latency {U1/U2/U3/Z}
```

The following example removes only rise ideal latency information from a port named *A*.

```
prompt> remove_ideal_latency -rise {A}
```

SEE ALSO

remove_ideal_network(2)
remove_ideal_transition(2)
report_ideal_network(2)
set_ideal_network(2)
set_ideal_latency(2)

remove_ideal_network

Removes sources of ideal networks in the current design. Cells and nets in the transitive fanout of the specified objects are no longer treated as ideal.

SYNTAX

status **remove_ideal_network**
-all | *object_list*

Data Types

object_list collection

ARGUMENTS

-all

Removes the ideal network for all objects in the current design.

object_list

Specifies a list of ideal sources. The source objects may be ports or pins of leaf cells at any hierarchical level of the design. If nets are specified in the *object_list* option, then all of the nets' global driver ideal source pins that were set with the **-no_propagate** option are removed.

DESCRIPTION

This command removes the ideal network property from a set of ports or pins in the design that were previously marked as ideal sources using the **set_ideal_network** command. You specify only the source of the network. The ideal network is automatically updated. The criteria for propagating the ideal property are detailed in the **set_ideal_network** man page.

All ideal networks are removed using the **reset_design** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example sets up an ideal network on objects in the design CLOCK_GEN and then removes part of the network:

```
prompt> current_design CLOCK_GEN
prompt> set_ideal_network {port1 port2}
prompt> remove_ideal_network port2
```

The following example creates an ideal network and then removes part of the ideal network.

```
prompt> current_design {TOP}
prompt> set_ideal_network {IN1 IN2}
prompt> remove_ideal_network {IN1}
```

The following example removes an ideal network that was set on a net.

```
prompt> set_ideal_network -no_propagate {netB}
Warning: Transferring ideal net attribute onto driver pin 'AN2/Z' of net 'netB'. (UITE-450)
prompt> remove_ideal_network {netB}
```

SEE ALSO

remove_ideal_latency(2)
remove_ideal_transition(2)
reset_design(2)
set_ideal_network(2)

remove_ideal_transition

Removes ideal transition values from the specified objects.

SYNTAX

int **remove_ideal_transition**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-all]
[-rise] [-fall]
[-min] [-max]
object_list

list *object_list*

ARGUMENTS

-modes *mode_list*

Specifies the modes for which to remove the ideal transition value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is de-annotated separately. If this option is not given, the command removes transition for every scenario that is active with the specified corner(s). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to remove the ideal transition value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is de-annotated separately. If this option is not given, the command removes transition for every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios for which to remove the ideal transition value. Each of the specified scenarios is de-annotated separately. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command applies to the current scenario only.

-all

Removes the ideal transition time from every pin in the current design. If you specify **-all**, you cannot specify any other arguments.

-rise

Removes the rise ideal transition time. If you do not specify the **-rise** or **-fall** options, both values are removed.

-fall

Removes the fall ideal transition time. If you do not specify the **-rise** or **-fall** options, both values are removed.

-min

Removes the minimum ideal transition time. If you do not specify the **-min** or **-max** options, both values are removed.

-max

Removes the maximum ideal transition time. If you do not specify the **-min** or **-max** options, both values are removed.

object_list

Specifies a list of ports or pins from which to remove ideal transition.

DESCRIPTION

Removes the user-specified ideal transition that was previously set by the **set_ideal_transition** command from specified objects in the *object_list* option.

To list ideal transition values, use the **report_ideal_network** command.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes ideal transition from the output pin named *Z* of cell instance *U1/U2/U3*.

```
prompt> remove_ideal_transition {U1/U2/U3/Z}
```

The following example removes only rise ideal transition information from a port named *A*.

```
prompt> remove_ideal_transition -rise {A}
```

SEE ALSO

remove_ideal_latency(2)
remove_ideal_network(2)
report_ideal_network(2)
set_ideal_network(2)
set_ideal_transition(2)

remove_ignored_layers

Removes ignored routing layers in congestion analysis and RC estimation. This command can also remove the design minimum and maximum routing layers if those layers have been already set.

SYNTAX

status **remove_ignored_layers**
[-min_routing_layer]
[-max_routing_layer]
[-all]
[-rc_congestion_ignored_layers *tech_layer_list*]

Data Types

tech_layer_list list

ARGUMENTS

-min_routing_layer

Removes the setting for the design minimum routing layer.

-max_routing_layer

Removes the setting for the design maximum routing layer.

-all

Removes all RC congestion ignored layers between design minimum and maximum routing layer inclusive.

-rc_congestion_ignored_layers *tech_layer_list*

Lists the routing layers that should no longer be ignored during congestion analysis and RC estimation.

DESCRIPTION

This command removes the ignored setting from some or all routing layers. You can specify routing layers to be ignored during congestion analysis and RC estimation.

Note that congestion analysis and RC estimation require that at least one horizontal and one vertical layer not be ignored.

You can also use this command to remove the settings for the design minimum and maximum routing layers.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command removes all RC congestion ignored layers between design minimum and maximum routing layer inclusive.

prompt> **remove_ignored_layers -all**

SEE ALSO

report_ignored_layers(2)
set_ignored_layers(2)

remove_individual_pin_constraints

Removes the existing individual pin constraints from nets and pins.

SYNTAX

string **remove_individual_pin_constraints**
[-nets *net_collection*]
[-pins *pin_collection*]
[-ports *port_collection*]

Data Types

net_collection	collection
pin_collection	collection
port_collection	collection

ARGUMENTS

-nets *net_collection*

Specifies the collection of nets from which to remove pin physical constraints.

-pins *pin_collection*

Specifies the collection of pins from which to remove pin physical constraints.

-ports *port_collection*

Specifies the collection of ports from which to remove pin physical constraints.

DESCRIPTION

This command removes all existing pin constraints from nets, pins and ports. If you do not specify any arguments, this command removes all pin constraints for all pins, nets and ports.

The **set_editability** command can enable or disable the **remove_individual_pin_constraints** command for specified blocks or levels of physical hierarchy. This command has no effect on the blocks that are disabled with the **set_editability** command.

EXAMPLES

The following example removes the pin constraints for all pins, nets and ports.

```
prompt> remove_individual_pin_constraints
```

SEE ALSO

report_individual_pin_constraints(2)
set_individual_pin_constraints(2)
set_editability(2)

remove_input_delay

Removes input delay information from ports or pins.

SYNTAX

string **remove_input_delay** [-clock *clock_name*]
[-clock_fall]
[-level_sensitive]
[-rise]
[-fall]
[-max]
[-min]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
port_pin_list

Data Types

clock_name list
mode_list list
corner_list list
scenario_list list
port_pin_list list

ARGUMENTS

-clock *clock_name*

Relative clock; '' for no clock. Use this option to remove only input delay relative to one clock.

-clock_fall

Delay is relative to falling edge of clock.

-level_sensitive

Delay is from level-sensitive latch.

-rise

Removes rising input delay.

-fall

Removes falling input delay.

-max

Removes maximum input delay.

-min

Removes minimum input delay.

-modes *mode_list*

Specifies the scenarios that input delays will be removed from. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option cannot be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners *corner_list*

Specifies the scenarios that input delays will be removed from. If this option is given, all scenarios of the given corners and the current mode will be used. The **-scenarios** option cannot be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios *scenario_list*

Specifies the scenarios that input delays will be removed from. The **-modes** or **-corners** option cannot be given with this option.

port_pin_list

Specifies a list of ports and pins.

DESCRIPTION

Removes input delay information from ports or pins in the current design. Input delay is specified with the **set_input_delay** command. To show input delays on ports, use **report_ports -input_delay**. The default is to remove all input delay information in the *port_pin_list*.

Input delay is managed on a per-scenario basis. For designs with multiple scenarios, you can specify scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command is applied to the current scenario. (It will be an error if there is no current scenario). If the **-scenarios** option is given, the command will be applied to all of the specified scenarios. If the **-modes** option is given, the command will be applied to all scenarios of the specified modes. (It will be an error if none of the specified modes have any scenarios.) If the **-corners** option is given, the command will be applied to all scenarios of the specified corners and the current mode. (It will be an error if none of the specified corners have any scenarios with the current mode.) If both the **-modes** and **-corners** options are given, the command will be applied to all scenarios of the specified corners and the specified modes. (It will be an error there are no such scenarios.) It is an error to give **-scenarios** with **-modes** or **-corners**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes all input delay from ports IN*.

prompt> **remove_input_delay [get_ports IN*]**

The following example removes input delay relative to CLK rise edge from port DATA[5].

prompt> **remove_input_delay -clock CLK { DATA[5] }**

SEE ALSO

remove_output_delay(2)
report_ports(2)
set_input_delay(2)
set_output_delay(2)

remove_io_filler_cells

Removes I/O filler cells from the current design.

SYNTAX

int **remove_io_filler_cells**
[-cells *cell_list*]

Data Types

cell_list list

ARGUMENTS

-cells *cell_list*

Specifies a list of I/O filler cells to remove. The list can contain I/O filler cell names, patterns, or collections. A collection can be specified by using the **get_cells** command. If no *cell_list* is specified the command removes all the I/O filler cells in the design.

DESCRIPTION

This command removes I/O filler cells from the current design, and returns the number of filler cells removed. An error will occur if you specify an incorrect set of cells or I/O filler cells. The removal of a filler cell, which is a member of user defined signal I/O constraints, will result in a warning.

EXAMPLES

The following example removes I/O filler cells named "io_filler1" and "io_filler2".

```
prompt> remove_io_filler_cells -cells { io_filler1 io_filler2 }
2
```

The following example removes I/O fillers that match the pattern "io_filler*".

```
prompt> remove_io_filler_cells -cells {io_filler*}
3
```

The following example removes all I/O filler cells in the design.

```
prompt> remove_io_filler_cells
5
```

SEE ALSO

create_io_filler_cells(2)

remove_io_guides

Removes I/O guides from the current design.

SYNTAX

int **remove_io_guides**
[-force]
[-verbose]
[-all]
[*io_guide_list*]

Data Types

io_guide_list list

ARGUMENTS

-force

Removes the I/O guide, even if the I/O guide object is locked. If this option is not specified, the command fails and generates a Tcl error if any of the specified objects is locked.

-verbose

Displays additional debugging information.

-all

Removes all I/O guides from the current design.

io_guide_list

Specifies a list of I/O guides to remove. The list can contain I/O guide names, patterns, or collections. A collection can be specified by using the **get_io_guides** command.

DESCRIPTION

This command removes I/O guide constraints from the current design. The number of removed I/O guides are returned. The command returns the number of I/O guides that were removed.

EXAMPLES

The following example removes I/O guides named "io_guide1" and "io_guide2".

```
prompt> remove_io_guides { io_guide1 io_guide2 }
2
```

The following example removes I/O guides that match the pattern "io_guide*".

```
prompt> remove_io_guides io_guide*
3
```

The following example removes all I/O guides in the design.

```
prompt> remove_io_guides -all
5
```

SEE ALSO

add_to_io_guide(2)
create_io_guide(2)
get_io_guides(2)
remove_from_io_guide(2)
report_io_guides(2)

remove_io_rings

Removes I/O rings from the current design.

SYNTAX

int **remove_io_rings**
[-force]
[-verbose]
[-all]
[*io_ring_list*]

Data Types

io_ring_list list

ARGUMENTS

-force

Removes the I/O ring, even if the object is locked. If this option is not provided and any of the specified objects has locked status, command fails and generates a Tcl error.

-verbose

Displays verbose debugging information.

-all

Removes all I/O rings from the current design.

io_ring_list

Specifies a list of I/O rings to remove. The list can contain I/O ring names, patterns, or collections. A collection can be specified with the **get_io_rings** command.

DESCRIPTION

This command removes I/O ring constraints from the current design. The command returns the number of removed I/O rings, or a Tcl error if there is a command syntax error.

EXAMPLES

The following example removes I/O rings named "io_ring1" and "io_ring2".

prompt> **remove_io_rings { io_ring1 io_ring2 }**

SEE ALSO

add_to_io_ring(2)
create_io_ring(2)
get_io_rings(2)
remove_from_io_ring(2)
report_io_rings(2)

remove_ivm

Reads via variation file(.ivm) for current desgin.

SYNTAX

status **remove_ivm**
[-corners *corners*]

Data Types

corners list

ARGUMENTS

-corners *corners*

Specifies the list of corners for which to apply the extraction options. If corners are not specified then the ivm file will be applied to all corners.

DESCRIPTION

This command removes via variation from the current design.

EXAMPLES

The following example removes via variation for specified two corners.

prompt> **remove_ivm -corners {FuncMode_max.SS.081v.125c_FuncCmax.corner FuncMode_min.FF.099v.125c_FuncCmax.**

SEE ALSO

report_ivm(2)
read_ivm(2)
write_ivm(2)

write_parasitics(2)

remove_keepout_margins

Removes the specified keepout margins.

SYNTAX

integer **remove_keepout_margins**
keepout_list

Data Types

keepout_list collection

ARGUMENTS

keepout_list

Specifies the keepout margins to remove. You can specify a single object or a collection of keepout margins.

This is a required argument.

DESCRIPTION

This command removes the specified keepout margins. If you delete a routing-blockage keepout margin, the command also deletes the corresponding layers in the owner object.

The command returns the number of keepout margins that were successfully deleted.

EXAMPLES

The following command removes all the keepouts in the current block:

prompt> **remove_keepout_margins [get_keepout_margins]**

SEE ALSO

create_keepout_margin(2)
get_keepout_margins(2)
report_keepout_margins(2)

remove_layer_map_file

Removes a layer mapping file from a library.

SYNTAX

status **remove_layer_map_file**
-format gds | starrc
[-library *library_name*]

Data Types

library_name string

ARGUMENTS

-format gds | **starrc**

Specifies the format of the layer mapping file, **gds** for GDS in/out or **starrc** for StarRC.

-library_name ***library_name***

Specifies the name of the library in which to store the layer mapping file. If not specified, the command applies to the current open library. This option is not valid for library shell. Library shell uses current open library only.

DESCRIPTION

This command removes a layer mapping file from a library.

EXAMPLES

The following example removes the GDS layer mapping file from current open library:

```
prompt> remove_layer_map_file \
    -format gds
```

SEE ALSO

set_layer_map_file(2)

remove_layers

Removes the given layer objects from the technology data.

SYNTAX

collection **remove_layers**
[-tech *tech*]
[-library *library*]
patterns

Data Types

tech string
library string
patterns string or collection

Data Types

patterns Collection or regexp pattern of layers to remove

ARGUMENTS

-tech *tech*

Specifies the technology data for finding the layer objects; the default is the current library's technology data.

-library *library*

Specifies the library for finding the layer objects; the default is the current library.

patterns

Specifies the collection of layers to be removed.

DESCRIPTION

This command removes the specified layers from the library's technology data. Objects on those layers, if any, will become unbound. If a layer is later re-added to the technology data, the unbound objects are re-bound to that layer.

RETURN VALUE

This command returns the number of layers removed. or an empty string if the command fails. If a command syntax error exists,

the command returns a TCL_ERROR.

EXAMPLES

The following example removes a layer named 'M1' from the technology data of the current library.

```
prompt> remove_layers M1
{1}
```

SEE ALSO

create_layer(2)
get_layers(2)
report_unbound(2)

remove_libcell_subset

Removes library cell subset family constraints from specified cells or removes a library cell subset family defined by the **define_libcell_subset** command.

SYNTAX

status **remove_libcell_subset**
[-object_list *cells*]
[-family_name *name*]

Data Types

cells list
name string

ARGUMENTS

-object_list *cells*

Specifies the cells from which to remove the library cell subset family. The cells must be unmapped sequential cells. You must specify either **-object_list** or **-family_name**. If neither **-object_list** nor **-family_name** is specified, an error occurs and the command quits.

-family_name *name*

Specifies the library cell subset family to be removed. You must specify either **-family_name** or **-object_list**. If neither **-family_name** nor **-object_list** is specified, an error occurs and the command quits.

DESCRIPTION

This command removes the library cell subset family specified by the **set_libcell_subset** command from the cells to be optimized, or it removes a library cell subset family defined by the **define_libcell_subset** command.

EXAMPLES

The following example removes the library cell subset family from the u1 sequential cell:

prompt> **remove_libcell_subset -object_list [get_cells u1]**

The following example removes the special_flops library cell subset family:

prompt> **remove_libcell_subset -family_name special_flops**

SEE ALSO

remove_libcell_subset(2)
report_libcell_subset(2)
set_libcell_subset(2)
define_libcell_subset(2)

remove_license

Removes a license of the feature.

SYNTAX

status **remove_licenses**
feature_name
[-keep *num_licenses*]

Data Types

feature_name	string
num_licenses	integer

ARGUMENTS

feature_name

Specifies the feature name to remove.

-keep *num_licenses*

Specifies the number of license features to be retained. If this option is not specified, all licenses of this feature are checked in.

DESCRIPTION

This command removes the specified license feature.

The **-keep** option keeps the specific number of licenses checked out and remaining are removed.

The **list_licenses** command provides a list of the features that are currently checked out.

The command **remove_licenses** will not be allowed while there are background jobs from redirect -bg running.

EXAMPLES

The following example removes all the licenses of the specified feature:

```
prompt> remove_licenses Fusion-Compiler-FE
```

The following example removes some, but not all:

```
prompt> list_licenses

Licenses in use:
        Fusion-Compiler-BE (4)
1
prompt> remove_licenses -keep 2 Fusion-Compiler-BE
1
prompt> list_licenses

Licenses in use:
        Fusion-Compiler-BE (2)
1
```

SEE ALSO

check_license(2)
list_licenses(2)
get_licenses(2)

remove_licenses

Removes a license of the feature.

SYNTAX

status **remove_licenses**
feature_name
[-keep *num_licenses*]

Data Types

feature_name	string
num_licenses	integer

ARGUMENTS

feature_name

Specifies the feature name to remove.

-keep *num_licenses*

Specifies the number of license features to be retained. If this option is not specified, all licenses of this feature are checked in.

DESCRIPTION

This command removes the specified license feature.

The **-keep** option keeps the specific number of licenses checked out and remaining are removed.

The **list_licenses** command provides a list of the features that are currently checked out.

The command **remove_licenses** will not be allowed while there are background jobs from redirect -bg running.

EXAMPLES

The following example removes all the licenses of the specified feature:

```
prompt> remove_licenses Fusion-Compiler-FE
```

The following example removes some, but not all:

```
prompt> list_licenses

Licenses in use:
        Fusion-Compiler-BE (4)
1
prompt> remove_licenses -keep 2 Fusion-Compiler-BE
1
prompt> list_licenses

Licenses in use:
        Fusion-Compiler-BE (2)
1
```

SEE ALSO

check_license(2)
list_licenses(2)
get_licenses(2)

remove_macro_constraints

Removes macro placement constraints.

SYNTAX

status **remove_macro_constraints**
[-allowed_orientations]
[-preferred_location]
[-alignment_grid]
[-alignment_point]
[-align_pins_to_tracks]
[-alignment_orientation_set all | R0 | R90]
[*hard_macro_list*]

Data Types

hard_macro_list list

ARGUMENTS

-allowed_orientations

Removes the allowed orientation constraints.

-preferred_location

Removes the preferred macro location constraints.

-alignment_grid

Removes the alignment grid and alignment point constraints.

-alignment_point

Removes the alignment point constraints.

-align_pins_to_tracks

Removes the align pins to tracks setting.

-alignment_orientation_set all| R0 | R90

Removes the alignment constraints for the specified orientation. By default, the all argument is used.

hard_macro_list

Specifies the list of hard macros on which to remove constraints. If not specified, all hard macros are processed when removing

constraints.

DESCRIPTION

This command removes various macro constraints set by the **set_macro_constraints** command.

EXAMPLES

The following command removes the allowed orientation and preferred location constraints from all hard macros.

```
prompt> remove_macro_constraints -allowed_orientations -preferred_location
1
```

SEE ALSO

create_placement(2)
report_macro_constraints(2)
set_macro_constraints(2)

remove_macro_groups

Remove macro groups.

SYNTAX

status **remove_macro_groups**
object object_list

Data Types

object collection of objects

ARGUMENTS

object collection_of_macro_group

Specifies macro groups to be removed

DESCRIPTION

This command remove specified macro groups.

EXAMPLES

The following command remove macro groups G1 and G2

prompt> **remove_macro_groupss** {G1 G2}

SEE ALSO

create_macro_group(2)

remove_macro_relative_location

Removes the previously set constraint which place a hard macro or a macro array relative to an anchor object.

SYNTAX

status **remove_macro_relative_location**
[*target_list*]
[-hierarchical]

Data Types

target_list list

ARGUMENTS

target_list

Specifies the list of hard macros or macro array edit groups on which to remove relative location constraints.

-hierarchical

Remove macro relative location constraints in current block and child block. Default only remove constraints in current block.

DESCRIPTION

This command removes specified hard macro or macro array constraints set by the **set_macro_relative_location** command.

EXAMPLES

The following command removes the relative location constraint set on hard macro cmem0/dtags0_0/x0_t0.

prompt> **remove_macro_relative_location [get_cells {cmem0/dtags0_0/x0_t0}]**
1

SEE ALSO

create_macro_relative_location_placement(2)
create_placement(2)
derive_macro_relative_location(2)
report_macro_relative_location(2)
set_macro_relative_location(2)
write_macro_relative_location(2)

remove_macros_from_group

Remove macros from their belonging macro groups.

SYNTAX

Status **remove_macros_from_group**
objects object_list

ARGUMENTS

objects object_list

Specifies collection of macros.

DESCRIPTION

This command remove macros from the macro group they belong to. If a macro belongs to a macro array, the whole array will be removed from the macro group.

EXAMPLES

The following command remove macros M1, M2 from their macro groups.

prompt> **remove_macros_from_groups** {M1 M2}

SEE ALSO

create_macro_groups(2)
add_macro_to_group(2)

remove_matching_types

Removes the matching type.

SYNTAX

int **remove_matching_types**
[-verbose]
[-all]
[*matching_type_list*]

Data Types

matching_type_list list

ARGUMENTS

-verbose

Displays verbose deletion information.

-all

Removes all matching types from the current design.

matching_type_list

Specifies a list of matching types to remove. The list may contain matching type names, patterns, or collections. A collection may be specified by using the **get_matching_types** command.

DESCRIPTION

This command removes matching types from the current design. The number of removed matching types are returned.

EXAMPLES

The following example removes matching types named "myMatchingType1" and "myMatchingType2".

```
prompt> remove_matching_types { myMatchingType1 myMatchingType2 }
2
```

The following example removes matching types that match the pattern "myMatchingType*".

```
prompt> remove_matching_types myMatchingType*
3
```

The following example removes all matching types in the design.

```
prompt> remove_matching_types -all
5
```

SEE ALSO

create_matching_type(2)
get_matching_types(2)
add_to_matching_type(2)
remove_from_matching_type(2)
report_matching_types(2)

remove_max_capacitance

Removes maximum capacitance limits from ports, lib_pins, cell instance pins, clocks or designs.

SYNTAX

string **remove_max_capacitance**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
object_list

Data Types

mode_list list
corner_list list
scenario_list list
object_list list

ARGUMENTS

-modes *mode_list*

Specifies the scenarios that the capacitance limit will be removed from. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option cannot be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners *corner_list*

Specifies the scenarios that the capacitance limit will be removed from. If this option is given, all scenarios of the given corners and the current mode will be used. The **-scenarios** option cannot be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios *scenario_list*

Specifies the scenarios that the capacitance limit will be removed from. The **-modes** or **-corners** option cannot be given with this option.

object_list

Provides a list of ports, lib_pins, cell instance pins, clocks or designs from which to remove maximum capacitance limits.

DESCRIPTION

Removes maximum capacitance limits from pins, ports, lib_pins, clocks or designs. A maximum capacitance limit is specified with the **set_max_capacitance** command. The command removes all limits set with or without **-corners** option of the **set_max_capacitance** command.

For lib_pin objects, it removes design specific overrides and the max capacitance values from the library will be used after this command.

The **report_ports -design_rule** command shows port maximum capacitance limits. The **report_design** command shows the default maximum capacitance setting for the scenarios of the current design.

Maximum capacitance limits are managed on a per-scenario basis. For designs with multiple scenarios, you can specify scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command is applied to the current scenario. (It will be an error if there is no current scenario). If the **-scenarios** option is given, the command will be applied to all of the specified scenarios. If the **-modes** option is given, the command will be applied to all scenarios of the specified modes. (It will be an error if none of the specified modes have any scenarios.) If the **-corners** option is given, the command will be applied to all scenarios of the specified corners and the current mode. (It will be an error if none of the specified corners have any scenarios with the current mode.) If both the **-modes** and **-corners** options are given, the command will be applied to all scenarios of the specified corners and the specified modes. (It will be an error there are no such scenarios.) It is an error to give **-scenarios** with **-modes** or **-corners**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes the maximum capacitance limit on ports "OUT*".

```
prompt> remove_max_capacitance [get_ports "OUT*"]
```

The following example removes the maximum capacitance limit on the current scenario.

```
prompt> remove_max_capacitance [current_design]
```

SEE ALSO

current_design(2)
get_ports(2)
remove_max_fanout(2)
remove_max_transition(2)
remove_min_capacitance(2)
report_design(2)
report_ports(2)
set_max_capacitance(2)

remove_max_fanout

NOTE

remove_max_fanout

Command not supported

remove_max_lvth_percentages

This command removes the constraint for maximum allowable percentage of low-Vth cells.

SYNTAX

status **remove_max_lvth_percentages**

DESCRIPTION

This command removes the maximum allowable percentage of low-Vth cells, a value which is a soft constraint set using set_max_lvth_percentage cmd. Optimization tries to meet timing and other constraints while using no more than the specified percentage of low-Vth cells, to control leakage power. But this is a soft constraint and Opto does not sacrifice TNS convergence to adhere to this percentage. After removal of this constraint, there is no soft target for lvt cells. Note that the percentage is computed based on cell count.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example removes the specified limit on low-Vth cells

prompt> **remove_max_lvth_percentages**

SEE ALSO

set_max_lvth_percentage(2)
set_threshold_voltage_group_type(2)

remove_max_time_borrow

Removes time borrow limit for latches.

SYNTAX

string **remove_max_time_borrow**
object_list
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]

Data Types

object_list list
mode_list list
corner_list list
scenario_list list

ARGUMENTS

object_list

Lists clocks, cells, data pins, or clock (enable) pins. If you specify a cell, all enable pins on that cell are affected.

-modes ***mode_list***

Specifies the scenarios that the time borrow limit will be removed from. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option cannot be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners ***corner_list***

Specifies the scenarios that the time borrow limit will be removed from. If this option is given, all scenarios of the given corners and the current mode will be used. The **-scenarios** option cannot be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios ***scenario_list***

Specifies the scenarios that the time borrow limit will be removed from. The **-modes** or **-corners** option cannot be given with this option.

DESCRIPTION

Removes maximum time borrow limit on objects. Time borrowing limits are specified with the **set_max_time_borrow** command. When the limit is removed, full time borrowing is allowed on level-sensitive latches.

Max_time_borrow values are managed on a per-scenario basis. For designs with multiple scenarios, you can specify scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command is applied to the current scenario. (It will be an error if there is no current scenario). If the **-scenarios** option is given, the command will be applied to all of the specified scenarios. If the **-modes** option is given, the command will be applied to all scenarios of the specified modes. (It will be an error if none of the specified modes have any scenarios.) If the **-corners** option is given, the command will be applied to all scenarios of the specified corners and the current mode. (It will be an error if none of the specified corners have any scenarios with the current mode.) If both the **-modes** and **-corners** options are given, the command will be applied to all scenarios of the specified corners and the specified modes. (It will be an error there are no such scenarios.) It is an error to give **-scenarios** with **-modes** or **-corners**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes the maximum time borrow limit on clock PHI1.

```
prompt> remove_max_time_borrow [get_clocks PHI1]
```

The following example removes the maximum time borrow limit on all the pins of cells matching U1/latch*.

```
prompt> remove_max_time_borrow [get_cells U1/latch*]
```

SEE ALSO

set_max_time_borrow(2)

remove_max_transition

Removes maximum transition limits from ports, library cell pins, cell instance pins, clocks or designs.

SYNTAX

string **remove_max_transition**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
object_list

Data Types

mode_list list
corner_list list
scenario_list list
object_list list

ARGUMENTS

-modes *mode_list*

Specifies the scenarios from which to remove the maximum transition limit. If this option is given, all scenarios of the specified modes will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners *corner_list*

Specifies the scenarios from which to remove the max transition limit. If this option is given, all scenarios of the specified corners and the current mode will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios *scenario_list*

Specifies the scenarios from which to remove the max transition limit. The **-modes** or **-corners** option must not be specified with this option.

object_list

Specifies the cell instance pins, ports, library cell pins, clocks or designs from which to remove maximum transition limits.

DESCRIPTION

This command removes maximum transition limits from pins, ports, library cell pins, clocks or designs. A maximum transition limit is specified with the **set_max_transition** command. The command removes all limits set with or without **-corners** option of **set_max_transition**.

For library cell pin objects, this command removes design-specific overrides. The max transition values from the library will be used after this command.

The **report_ports -design_rule** command shows maximum transition limits for the port. The **report_design** command shows the default maximum transition setting for the scenarios of the current design.

Maximum transition limits are managed on a per-scenario basis. For designs with multiple scenarios, you can specify scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command applies to the current scenario. The tool issues an error if there is no current scenario. If the **-scenarios** option is specified, the command applies to all of the specified scenarios.

If the **-modes** option is specified, the command applies to all scenarios of the specified modes. The tool issues an error if none of the specified modes have any scenarios.

If the **-corners** option is specified, the command applies to all scenarios of the specified corners and the current mode. The tool issues an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are specified, the command applies to all scenarios of the specified corners and the specified modes. The tool issues an error if there are no such scenarios. It is an error to specify **-scenarios** with **-modes** or **-corners**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes the maximum transition limit on ports "OUT*".

```
prompt> remove_max_transition [get_ports "OUT*"]
```

The following example removes the maximum transition limit on the current scenario.

```
prompt> remove_max_transition [current_design]
```

SEE ALSO

current_design(2)
get_ports(2)
report_design(2)
report_ports(2)
set_max_capacitance(2)
set_max_transition(2)

remove_min_capacitance

Removes minimum capacitance limits from ports, lib_pins or designs.

SYNTAX

string **remove_min_capacitance**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
object_list

Data Types

mode_list list
corner_list list
scenario_list list
object_list list

ARGUMENTS

-modes *mode_list*

Specifies the scenarios that the min capacitance limit will be removed from. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option cannot be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners *corner_list*

Specifies the scenarios that the min capacitance limit will be removed from. If this option is given, all scenarios of the given corners and the current mode will be used. The **-scenarios** option cannot be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios *scenario_list*

Specifies the scenarios that the min capacitance limit will be removed from. The **-modes** or **-corners** option cannot be given with this option.

object_list

Lists the ports, lib_pins or designs from which to remove minimum capacitance limits.

DESCRIPTION

Removes minimum capacitance limits from ports or designs. A minimum capacitance limit is specified with the **set_min_capacitance** command. The command removes all limits set with or without **-corners** option of set_min_capacitance command.

For lib_pin objects, it removes design specific overrides and the min capacitance values from the library will be used after this command.

The **report_ports -design_rule** command shows port minimum capacitance limits. The **report_design** command shows the default minimum capacitance setting for the scenarios of the current design.

Minimum capacitance limits are managed on a per-scenario basis. For designs with multiple scenarios, you can specify scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command is applied to the current scenario. (It will be an error if there is no current scenario). If the **-scenarios** option is given, the command will be applied to all of the specified scenarios. If the **-modes** option is given, the command will be applied to all scenarios of the specified modes. (It will be an error if none of the specified modes have any scenarios.) If the **-corners** option is given, the command will be applied to all scenarios of the specified corners and the current mode. (It will be an error if none of the specified corners have any scenarios with the current mode.) If both the **-modes** and **-corners** options are given, the command will be applied to all scenarios of the specified corners and the specified modes. (It will be an error there are no such scenarios.) It is an error to give **-scenarios** with **-modes** or **-corners**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes the minimum capacitance limit on ports "OUT*".

```
prompt> remove_min_capacitance [get_ports "OUT*"]
```

The following example removes the minimum capacitance limit on the current scenario.

```
prompt> remove_min_capacitance [current_design]
```

SEE ALSO

current_design(2)
get_ports(2)
remove_max_capacitance(2)
report_design(2)
report_ports(2)
set_max_capacitance(2)
set_max_transition(2)
set_min_capacitance(2)

remove_min_pulse_width

Removes a previously specified minimum pulse width constraint from specified clocks or clock pins.

SYNTAX

status **remove_min_pulse_width**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-low]
[-high]
[*object_list*]

Data Types

mode_list list
corner_list list
scenario_list list
object_list list

ARGUMENTS

-modes *mode_list*

Specifies the scenarios on which to set the input delay value. If this option is given, all scenarios of the given modes are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-corners *corner_list*

Specifies the scenarios on which to set the input delay value. If this option is given, all scenarios of the given corners and the current mode is used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-scenarios *scenario_list*

Specifies the scenarios on which to set the input delay value. The **-modes** or **-corners** option must not be given with this option.

-low

Removes the minimum pulse width constraint only for low clock pulses (high-low-high).

-high

Removes the minimum pulse width constraint only for high clock pulses (low-high-low).

If you do not specify the *-low* or *-high* option, the command removes the minimum pulse width constraint for all types of clock pulses.

object_list

Specifies a list of clocks or clock pins in the current design from which to remove the minimum pulse width constraint. If you specify a clock, all clock pins connected to the clock have their minimum pulse width constraints removed.

DESCRIPTION

The **remove_min_pulse_width** command removes pulse width checks previously specified by the **set_min_pulse_width** command for clock signals at clock pins of sequential devices. This command removes the constraints for current scenario.

To generate a report of pulse width constraints, use the **report_min_pulse_width** command.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes a minimum pulse width requirement previously set for clock CK1.

```
prompt> remove_min_pulse_width [get_clocks CK1]
```

The following example removes a minimum pulse width requirement previously set for pin reg1/CK.

```
prompt> remove_min_pulse_width reg1/CK
```

SEE ALSO

report_min_pulse_width(2)
set_min_pulse_width(2)

remove_missing_via_check_options

Removes missing via check options for the RedHawk Analysis Fusion check missing via analysis flow that were previously defined with the **set_missing_via_check_options** command.

SYNTAX

status **remove_missing_via_check_options**
[-redhawk_script_file]
[-exclude_stack_via]
[-check_valid_vias]
[-exclude_cell]
[-exclude_inst]
[-exclude_regions]
[-sort_by_hot_inst]
[-threshold]
[-min_width]
[-pitch]
[-ignore_inter_met]
[-toplayer]
[-bottomlayer]
[-gds]
[-all]

ARGUMENTS

-redhawk_script_file *redhawk_script_file*

Removes the setting to use an existing script file for missing via check.

-exclude_stack_via

Removes the setting to exclude stack vias from missing via check.

-check_valid_vias

Removes the setting to report if metal overlap can accommodate a valid via model.

-exclude_cell *exclude_cell_masters*

Removes the setting to exclude the area covered by all instances of the specified list of cells.

-exclude_inst *exclude_cell_instances*

Removes the setting to exclude the area covered by the specified cell instances.

-exclude_regions {{{x1 y1} {x2 y2}}}

Removes the setting to exclude from multiple rectangular regions.

-sort_by_hot_inst

Removes the setting to sort missing vias by voltage drop.

-threshold -1 | **<voltage_V>**

Removes voltage threshold that triggers a missing via.

-min_width ***min_width_value***

Removes the setting to ignore segments with less than specified width.

-pitch ***pitch_value***

Removes the specified pitch for missing vias on parallel wires crossing or touching GDS blocks.

-ignore_inter_met

Removes the setting to ignore wires on all layers between top layer and bottom layer.

-toplayer ***toplayer_name***

Removes the setting on the top layer.

-bottomlayer ***bottomlayer_name***

Removes the setting on the bottom layer.

-gds

Removes the setting to report missing vias both inside the GDS block region and in routes cross over GDS blocks.

-all

Removes all missing via setting.

EXAMPLES

The following example configures the current check missing via settings not to report if metal overlap can accommodate a valid via model.

prompt> **remove_missing_via_check_options -check_valid_vias**

SEE ALSO

analyze_rail(2)
set_missing_via_check_options(2)
report_missing_via_check_options(2)
set_app_options(2)

remove_modes

Removes modes in multi mode analysis.

SYNTAX

remove_modes -all | *mode_list*

ARGUMENTS

-all

Specifies to remove all modes in the current design.

mode_list

A list of unique strings used to identify each mode.

DESCRIPTION

This command removes the specified modes from memory. To list the modes in the design, use the **all_modes** command.

The command returns the number of modes that were removed.

EXAMPLES

In the following example, two modes named mode1 and mode2 are removed.

```
prompt> remove_modes {mode1 mode2}
1
```

SEE ALSO

create_mode(2)
all_modes(2)

remove_modules

Removes one or more unused modules from a design.

SYNTAX

collection_handle **remove_modules**
  [-design *design*]
  modules

modules list

ARGUMENTS

-design *design*

Specifies the design to remove the modules from. If no design is specified, the modules are removed from the current_design.

modules

Specifies the names of the modules to be removed from the design. Each module name must be unique.

DESCRIPTION

This command removes modules from a design. A module is a definition of logical hierarchy within a physical design block. The module must not be in use, there must be no cells that are instances of the module. The top module of a design cannot be removed with **remove_modules**, use the **remove_block** command instead.

EXAMPLES

The following example removes a module named *FOO*.

prompt> **remove_modules FOO**

The following example removes modules named *HOP*, *SKIP*, and *JUMP*.

prompt> **remove_modules {HOP SKIP JUMP}**

SEE ALSO

current_design(2)
remove_blocks(2)
remove_modules(2)

remove_multisource_clock_sink_groups

Removes sink groups defined for multisource clock tap assignment

SYNTAX

status **remove_multisource_clock_sink_groups**
[*sink_groups*]

Data Types

sink_groups collection

ARGUMENTS

sink_groups

Specifies a list of sink groups to be removed. Sink groups are defined using the **create_multisource_clock_sink_group** command. This argument is optional. By default, the command will remove all defined sink groups.

DESCRIPTION

The **remove_multisource_clock_sink_group** command is used to remove sink groups defined for multisource clock tap assignment.

The argument **sink_groups** can be used to remove only the given list of sink groups.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This example removes all defined sink groups.

```
remove_multisource_clock_sink_groups
```

This example removes only sink group sink_group1.

```
remove_multisource_clock_sink_groups sink_group1
```

This example removes a collection of sink groups.

```
remove_multisource_clock_sink_groups [get_multisource_clock_Sink_groups sink_group*]
```

SEE ALSO

create_multisource_clock_sink_group(2)
report_multisource_clock_sink_groups(2)
get_multisource_clock_sink_groups(2)
add_to_multisource_clock_sink_group(2)
remove_from_multisource_clock_sink_group(2)
synthesize_multisource_clock_taps(2)
set_multisource_clock_tap_options(2)

remove_multisource_clock_subtree_constraints

Removes constraints for structural synthesis of multisource clock tree synthesis (CTS) subtrees.

SYNTAX

status **remove_multisource_clock_subtree_constraints**
[-names *name_list*]
[-clocks *clock_list*]
[-cells *cell_list*]
[-pins *pins_or_ports*]
[-target_level *levels*]
[-ignore_for_icg_reordering]

Data Types

name_list list of names
clock_list collection of clocks
cell_list collection of cells
pins_or_ports list or collection
levels integer

ARGUMENTS

-names *name_list*

Specifies the subtree sets for which to remove constraints on pins or cells. If neither this option nor the **-clocks** option are specified, constraints are removed for all subtree option sets. If only this option is specified, all structural subtree synthesis constraints are removed for the specified named subtree sets.

-clocks *clock_list*

Specifies the subtree sets for which to remove constraints on pins or cells. If neither this option nor the **-names** option are specified, constraints are removed for all subtree sets. If only this option is specified, all subtree synthesis constraints are removed for the specified clocks.

-cells *cell_list*

Specifies a collection of cells on which to remove constraints. An example constraint is "ignore for ICG reordering".

-pins *pins_or_ports*

Specifies a collection of pins on which to remove constraints. An example constraint is "target level".

-target_level *levels*

Specifies the exact number of target levels for any pin. Any existing pin constraint matching the number of target levels will be

removed. The minimum level value is 0 while the maximum is restricted to 100.

-ignore_for_icg_reordering

Specifies that the "ignore for ICG reordering" constraint is removed for cells specified by the **-cells** option. If the **-cells** option is not specified, the constraint is removed for all cells.

DESCRIPTION

The **remove_multisource_clock_subtree_constraints** command removes constraints on cells or pins; these constraints are read and honored by the structural multisource CTS synthesis using the **synthesize_multisource_clock_subtrees** command.

For constraint removal on pins, the **-target_level** option helps you to better control subtree logic level balancing.

For constraint removal on cells, the **-ignore_for_icg_reordering** option helps you to better control subtree synthesis transform ICG reordering.

Use the **set_multisource_clock_subtree_constraints** command to set clock subtree constraints; use the **report_multisource_clock_subtree_constraints** command to report clock subtree constraints; use the **synthesize_multisource_clock_subtrees** command to synthesize the subtrees based on the specified constraints.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This example removes all constraints on all clocks.

```
prompt> remove_multisource_clock_subtree_constraints
```

This example removes all constraints on the clock named "clk".

```
prompt> remove_multisource_clock_subtree_constraints \
  -clocks [get_clock clk]
```

This example removes constraints from pins reg1/ck and reg2/ck.

```
prompt> remove_multisource_clock_subtree_constraints \
  -clocks [get_clock clk] -pins [get_pins {reg1/ck reg2/ck}]
```

This example removes constraints from pins reg1/ck and reg2/ck with a target level of 5.

```
prompt> remove_multisource_clock_subtree_constraints \
  -clocks [get_clock clk] -pins [get_pins {reg1/ck reg2/ck}] \
  -target_level 5
```

This example removes constraints from all pins with the specific target level.

```
prompt> remove_multisource_clock_subtree_constraints \
  -clocks [get_clock clk] -target_level 4
```

This example removes constraints from specific cells:

```
prompt> remove_multisource_clock_subtree_constraints \
  -clocks [get_clock clk] -cells [get_cells {icg1 icg2}]
```

This example removes ignore_for_icg_reordering constraints from all cells.

```
prompt> remove_multisource_clock_subtree_constraints \
  -clocks [get_clock clk] -ignore_for_icg_reordering
```

SEE ALSO

remove_multisource_clock_subtree_options(2)
report_multisource_clock_subtree_constraints(2)
report_multisource_clock_subtree_options(2)
set_multisource_clock_subtree_constraints(2)
set_multisource_clock_subtree_options(2)
synthesize_multisource_clock_subtrees(2)

remove_multisource_clock_subtree_options

Removes structural multisource subtree synthesis constraints that were previously applied with **set_multisource_clock_subtree_options**

SYNTAX

status **remove_multisource_clock_subtree_options**
[-names *string_list*]
[-clocks *clock_list*]
[-driver_objects]
[-max_total_wire_delay]
[-corners *corner_list*]
[-dont_merge_cells *cell_list*]
[-target_level]

Data Types

string_list list
clock_list collection
corner_list collection
cell_list collection

ARGUMENTS

-names *string_list*

Specifies the subtree sets for which to remove constraints. If neither this option nor the **-clocks** option are specified, then constraints are removed for all subtree option sets. If only this option is specified, then all structural subtree synthesis constraints are removed for the specified named subtree sets.

This option can also be specified together with one or more of the **-driver_objects**, **-max_total_wire_delay**, or **-corners** options. Including these options clear only those constraints for the specified subtree sets.

-clocks *clock_list*

Specifies the subtree sets for which to remove constraints. If neither this option nor the **-names** option are specified, then constraints are removed for all subtree sets. If only this option is specified, then all subtree synthesis constraints are removed for the specified clocks.

This option can also be specified together with one or more of the **-driver_objects**, **-max_total_wire_delay**, or **-corners** options. Including these options clear only those constraints for the specified sets.

-driver_objects

Removes the driver object constraint settings. If you use **-driver_objects** without the **-names** or **-clocks** option, the driver objects

constraints for all subtree sets are removed. If you use the **-driver_objects** option with the **-names** or **-clocks** option, then the driver object constraints is cleared only for the specified subtree sets.

This option can be specified together with any of the **-max_total_wire_delay**, or **-corners** options to clear multiple constraint types in one command invocation.

-max_total_wire_delay

Removes the max total wire delay settings. If you use **-max_total_wire_delay** without the **-names** or **-clocks** option, the maximum total wire delay settings for all subtree sets is removed. If you use **-max_total_wire_delay** with the **-names** or **-clocks** option, then the maximum total wire delay settings is cleared only for the specified subtree sets.

If you use **-max_total_wire_delay** without the **-corners** option, then all maximum total wire delay settings are cleared for the subtree sets. If you use **-max_total_wire_delay** with the **-corners** option, then only the maximum total wire delay settings for the specified corners is cleared.

If the maximum total wire delay constraint is cleared for all corners for a particular option set, by default, the maximum total wire delay is unconstrained for that set of subtrees.

This option can be specified together with **-driver_objects** to clear multiple constraint types in one command invocation.

-corners ***corner_list***

Removes the maximum total wire delay settings for specified corners. you must use **-corners** together with the **-max_total_wire_delay** option, otherwise an error occurs.

If you use **-corners** without the **-names** or **-clocks** option, the maximum total wire delay constraints for the specified corners is cleared for all subtree sets. If you use **-corners** with the **-names** or **-clocks** option, the maximum total wire delay constraints for the specified corners is cleared only for the specified subtree sets.

-dont_merge_cells ***cell_list***

This option accepts a collection of cells which should get removed from the list of not to be merged cells of the option set.

-target_level

Removes the target level setting. If level balancing is still enabled for an option set, then level balancing falls back to automatically determining the number of levels.

DESCRIPTION

This command clears settings applied with the **set_multisource_clock_subtree_options** command. By default, this command operates on all subtree sets at one time. The scope of this command can be limited to just a subset of the subtrees by using the **-names** or **-clocks** option.

This command can clear two types of subtree set constraints:

(1) The driver objects for the subtrees (2) Maximum total wire delay constraints in one or more corners

If none of the **-driver_objects**, or **-max_total_wire_delay** options are specified, then the entire subtree sets are removed.

The **-corners** option can only be used together with the **-max_total_wire_delay** option to clear only the maximum total wire delay constraints for the specified corners. If the corners option is not used but **-max_total_wire_delay** is specified, then the maximum total wire delay constraints are cleared for all corners.

If all constraints for a subtree set are cleared, then the set itself is also removed. If only some of the constraints are cleared, then the constraint definition remain and the cleared settings revert to their defaults, which are described under each argument description

above.

EXAMPLES

This example clears all subtree synthesis sets.

```
prompt> remove_multisource_clock_subtree_options
```

This example clears all settings only for *subtree1*. The *subtree1* definition is removed, since all of its settings were cleared.

```
prompt> remove_multisource_clock_subtree_options -names subtree1
```

This example clears all settings of the subtree sets defined with respect to clock *clk*. The subtree set is removed, since all of its settings were cleared.

```
prompt> remove_multisource_clock_subtree_options -clocks [get_clock clk]
```

This example clears the max total wire delay settings for corners *c1* and *c2* for subtree set *subtree1* and *subtree2*.

```
prompt> remove_multisource_clock_subtree_options -names "subtree1 subtree2" \
  -max_total_wire_delay -corners [get_corners {c1 c2}]
```

This example clears the max total wire delay settings for corners *c1* and *c2* for subtree settings of clock *clk1* and *clk2*.

```
prompt> remove_multisource_clock_subtree_options -clocks [get_clocks {clk1 clk2}] \
  -max_total_wire_delay -corners [get_corners {c1 c2}]
```

SEE ALSO

report_multisource_clock_subtree_options(2)
set_multisource_clock_subtree_options(2)
synthesize_multisource_clock_subtrees(2)

remove_multisource_clock_tap_options

Removes multisource clock tree synthesis (CTS) tap assignment constraints that were previously applied with **set_multisource_clock_tap_options**

SYNTAX

status **remove_multisource_clock_tap_options**
[-names *string_list*]
[-clocks *clock_list*]
[-dont_merge_cells *cell_list*]
[-relax_split_restrictions_for_cells *cell_list*]

Data Types

string_list list
clock_list collection
cell_list collection

ARGUMENTS

-names *string_list*

Specifies the tap assignment option sets which should get removed. If neither this option nor the **-clocks** option are specified, then all tap assignment option sets are removed. If only this option is specified, then all option sets with the given names are removed.

-clocks *clock_list*

Specifies the tap assignment option sets for which to remove constraints. If neither this option nor the **-names** option are specified, then all tap assignment option sets are removed. If only this option is specified, then all option sets for the given clocks are removed.

-dont_merge_cells *cell_list*

This option accepts a collection of cells which should get removed from the list of not to be merged cells of the option set.

-relax_split_restrictions_for_cells *cell_list*

This option accepts a collection of cells which should get removed from the list of cells with user_dont_touch/user_size_only allowed for splitting during tap assignment with the option set.

DESCRIPTION

This command clears settings applied with the **set_multisource_clock_tap_options** command. By default, this command removes all option sets at one time. The scope of this command can be limited to just a subset of options by using the **-names** or **-clocks** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This example clears all subtree synthesis option sets.

```
prompt> remove_multisource_clock_tap_options
```

This example clears all settings only for *subtree1*. The *subtree1* definition is removed.

```
prompt> remove_multisource_clock_tap_options -names subtree1
```

This example clears all settings of the clock *clk*.

```
prompt> remove_multisource_clock_tap_options -clocks [get_clock clk]
```

SEE ALSO

report_multisource_clock_tap_options(2)
set_multisource_clock_tap_options(2)
synthesize_multisource_clock_taps(2)

remove_multisource_global_clock_trees

Removes clock cells and routes inserted by *synthesize_multisource_global_clock_trees*.

SYNTAX

status **remove_multisource_global_clock_trees**
[-nets *net_list*]
[-prefix *name*]

Data Types

net_list collection
name string

ARGUMENTS

-nets *net_list*

Only those cells are deleted which are in the forward cone of the pins driving the specified nets.

-prefix *name*

Only those cells are deleted which have a name starting exactly with the specified prefix. The passed name should not contain a wildcard.

DESCRIPTION

The *remove_multisource_global_clock_trees* command will delete cells and routes inserted by previous calls of command *synthesize_multisource_global_clock_trees*. With the two provided options the user can limit the set of cells being deleted. By using the option *-nets* only those cells are deleted which are in the forward cone of the pins driving the specified nets. With the option *-prefix* the set of removed cells can be limited to those which start exactly with the specified prefix. If needed both options can be specified.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example removes all cells inserted with calls to *synthesize_multisource_global_clock_trees* having a name starting with foo.

prompt> **remove_multisource_global_clock_trees -prefix foo**

The following example removes all cells inserted with calls to *synthesize_multisource_global_clock_trees* which are in the forward cone of the driver of net clk1 and have name starting with bar.

prompt> **remove_multisource_global_clock_trees -nets [get_nets clk1] -prefix bar**

SEE ALSO

synthesize_multisource_global_clock_trees(2)
create_clock_drivers(2)
remove_clock_drivers(2)

remove_net

Removes nets from the specified design.

SYNTAX

int **remove_nets**
[-design *design*]
[-force]
[-remove_shapes]
net_list | -all

net_list list

ARGUMENTS

-design *design*

Specifies the design in which to remove the net(s). If no design is specified, the net(s) are removed from the current design.

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-remove_shapes

Removes the shapes, including vias, owned by the nets being removed. When this option is not specified, the shapes remain in the design without an owner and may be reassigned to another net.

net_list

Specifies a list of nets to be removed from the design. Each net name in *net_list* must exist in the design.

You must specify *net_list* or **-all**. The *net_list* and **-all** options are mutually exclusive.

-all

Removes all nets in the design. The *net_list* and **-all** options are mutually exclusive.

DESCRIPTION

This command removes nets from the current design (unless otherwise specified by -design). Net connections to pins or ports are disconnected.

To create nets, use the **create_net** command.

EXAMPLES

The following example removes all nets specified by the *N** collection in the current design.

```
prompt> get_nets "*"
{"NET0", "NET1", "NET2", "NET3", "MY_CHECK", "PARITY"}

prompt> remove_nets "N*"
Removing net 'NET0' in design 'my_design'.
Removing net 'NET1' in design 'my_design'.
Removing net 'NET2' in design 'my_design'.
Removing net 'NET3' in design 'my_design'.

prompt> get_nets "*"
{"MY_CHECK", "PARITY"}
```

The following example removes all the remaining nets in the current design shown in the example given above.

```
prompt> get_nets "*"
{VSS out S in CLK VDD clkf clki0 clki1 clk din}

prompt> remove_nets -all
11

prompt> get_nets "*"
Warning: No net objects matched '*' (SEL-004)
Error: Nothing matched for collection (SEL-005)
```

SEE ALSO

create_net(2)
current_design(2)
get_nets(2)

remove_net_buses

Removes net buses from the current design.

SYNTAX

collection **remove_net_buses**
[-design *design*]
[-force]
[-block *block*]
[-cell *cell*]
[-all]
net_bus_list

Data Types

block string or collection
cell string or collection
net_bus_list string or collection

ARGUMENTS

-design *design*

Specifies the design in which to remove the net buses. If no design is specified, the net buses are removed from the current design.

-block *block*

Specifies the block where the net bus is to be removed. If specified, **remove_net_buses** has the same effect as **edit_module** in which the module is changed and changes will be on all module occurrences.

-cell *cell*

Specifies the cell where the net bus is to be removed. The net bus is removed on the cell's reference module. **remove_net_buses** will first uniquify the reference if needed, then remove the net bus. When not specified, the net bus is removed on the current module.

-force

Ignores the locked status of nets of the bus net. If this option is not specified and any of the affected objects has locked status, the command issues a Tcl error and exits.

-all

Removes all net buses.

net_bus_list

Specifies a list of net buses to remove.

RETURN VALUE

This command returns the removed net bus count, an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

DESCRIPTION

This command removes a list of net buses from the current design (unless otherwise specified by -design). Individual members of buses remain.

EXAMPLES

The following example removes net bus named *bus1*

```
prompt> remove_net_buses bus1
1
```

SEE ALSO

add_to_net_bus(2)
create_net_bus(2)
get_net_buses(2)
remove_from_net_bus(2)
report_net_buses(2)

remove_net_estimation_rules

Removes a net estimation rule.

SYNTAX

int **remove_net_estimation_rules**
[-cell *cell*]
[rules]

Data Types

rules net_estimation_rules
cell collection

ARGUMENTS

rules

Specifies the name of a net estimation rule or rules to remove. The rule must have been previously define by the **set_net_estimation_rule** command.

-cell *cell*

Specifies the physical cell from which to remove the net estimation rule.

By default, the current block is searched for the specified net estimation rule(s)..

DESCRIPTION

Removes net estimation rules as previously defined by the **set_net_estimation_rule** command.

Only rules which are not in use can be removed. Also, the default rule cannot be removed. The default rule has the name default.

NET ESTIMATION RULES AND HIERARCHY

Net estimation rules can be created in any physical block at any level of the hierarchy and multiple blocks can have different rules with the same name. However, rules are often stored/retrieved by name, and doing this can cause confusion as to which rule is being used by a given command. Generally, the "current" (top) block's list of rules is searched by commands such as *estimate_timing*. This means that if you define rule R1 in "top" and rule R1 in block "B1", if you make block B1 the current block, commands will use B1's R1. When you go back to top, top's R1 is used. To avoid confusion, use globally unique rule names and define them at the top - they will be propagated to child blocks during distributed commands as necessary. Rules in child blocks that have the same name as rules in top may also be overwritten during distributed commands.

EXAMPLES

The following example creates a rule named new_rule and removes it.

```
prompt> set_net_estimation_rule -parameter register_spacing -value 10 new_rule
prompt> remove_net_estimation_rules new_rule
(info) Removed net estimation rule new_rule.
1
```

The following example removes net estimation rules rule1 and rule2.

```
prompt> remove_net_estimation_rules {rule1 rule2}
(info) Removed net estimation rule rule1
(info) Removed net estimation rule rule2
1
```

SEE ALSO

get_net_estimation_rules(2)
report_net_estimation_rules(2)
set_net_estimation_rule(2)

remove_net_weight_effort

Remove net weight effort for coarse placement.

SYNTAX

status **remove_net_weight_effort**
[-nets *collection of nets*]

ARGUMENTS

-nets *collection of nets*

Removes the net weight effort for all nets, or for the nets specified.

DESCRIPTION

The **remove_net_weight_effort** command removes the net weight effort on nets for coarse placement. If no nets are specified, the net weight effort is cleared for all nets.

EXAMPLES

The following example removes the net weight effort for the net n123.

prompt> **remove_net_weight_effort -nets [get_nets n123]**

SEE ALSO

set_net_weight_effort(2)
report_net_weight_effort(2)

remove_nets

Removes nets from the specified design.

SYNTAX

int **remove_nets**
[-design *design*]
[-force]
[-remove_shapes]
net_list | -all

net_list list

ARGUMENTS

-design *design*

Specifies the design in which to remove the net(s). If no design is specified, the net(s) are removed from the current design.

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-remove_shapes

Removes the shapes, including vias, owned by the nets being removed. When this option is not specified, the shapes remain in the design without an owner and may be reassigned to another net.

net_list

Specifies a list of nets to be removed from the design. Each net name in *net_list* must exist in the design.

You must specify *net_list* or **-all**. The *net_list* and **-all** options are mutually exclusive.

-all

Removes all nets in the design. The *net_list* and **-all** options are mutually exclusive.

DESCRIPTION

This command removes nets from the current design (unless otherwise specified by -design). Net connections to pins or ports are disconnected.

To create nets, use the **create_net** command.

EXAMPLES

The following example removes all nets specified by the *N** collection in the current design.

```
prompt> get_nets "*"
{"NET0", "NET1", "NET2", "NET3", "MY_CHECK", "PARITY"}

prompt> remove_nets "N*"
Removing net 'NET0' in design 'my_design'.
Removing net 'NET1' in design 'my_design'.
Removing net 'NET2' in design 'my_design'.
Removing net 'NET3' in design 'my_design'.

prompt> get_nets "*"
{"MY_CHECK", "PARITY"}
```

The following example removes all the remaining nets in the current design shown in the example given above.

```
prompt> get_nets "*"
{VSS out S in CLK VDD clkf clki0 clki1 clk din}

prompt> remove_nets -all
11

prompt> get_nets "*"
Warning: No net objects matched '*' (SEL-004)
Error: Nothing matched for collection (SEL-005)
```

SEE ALSO

create_net(2)
current_design(2)
get_nets(2)

remove_noise_margin

Removes noise margin for a library pin, port, or pin.

SYNTAX

status **remove_noise_margin**
[-above]
[-below]
[-low]
[-high]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
object_list

Data Types

mode_list list
corner_list list
scenario_list list
object_list list

ARGUMENTS

-above

Removes the noise margin for above ground or power rail noise analysis region.

-below

Removes the noise margin for below ground or power rail noise analysis region.

-low

Removes the noise margin for ground rail noise.

-high

Removes the noise margin for power rail noise.

-modes *mode_list*

Specifies the scenarios on which to remove the noise margin value. If this option is given, all scenarios of the given modes are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-corners ***corner_list***

Specifies the scenarios on which to remove the noise margin value. If this option is given, all scenarios of the given corners and the current mode is used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-scenarios ***scenario_list***

Specifies the scenarios on which to remove the noise margin value. The **-modes** or **-corners** option must not be given with this option.

object_list

Specifies a list of library pins, ports, or pins.

DESCRIPTION

This command removes the noise margin information set by the set_noise_margin command.

For a library pin, this command is for current corner when without **-corners** option, and only supports **-corners** option to specify corners. For a design port or pin, this command is for current scenario when without **-scenarios**, **-corners** and **-modes** options, or using the three options to specify scenarios.

EXAMPLES

This example removes noise margin information for above the ground rail noise for pin A of library cell IV in the lsi_10k library:

prompt> **remove_noise_margin -above lsi_10k/IV/A**

SEE ALSO

set_noise_margin(2)

remove_objects

Removes a collection of database objects.

SYNTAX

int **remove_objects**
[-force]
object_list

Data Types

object_list collection

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

object_list

Specifies a list of objects to remove.

DESCRIPTION

This command removes a heterogenous collection of objects from the database, and returns the number of objects removed. For most object types, an error will occur if one or more of the objects could not be removed. The failure to remove a select few object types, such as voltage_areas and voltage_area_shapes when they are the respective defaults, will result in a warning.

EXAMPLES

The following removes a collection of cells and site rows:

```
prompt> set myobjs [get_cells U21*]
{U210 U211 U212 U213 U214 U215 U216 U217 U218 U219}
```

```
prompt> append_to_collection myobjs [get_site_rows ROW_5*]
{U210 U211 U212 U213 U214 U215 U216 U217 U218 U219 ROW_50 ROW_51 ROW_52 ROW_5}

prompt> remove_objects $myobjs
14
```

SEE ALSO

collections(2)

remove_ocvm

Removes advanced and parametric on-chip variation modeling (AOCVM / POCVM) information.

SYNTAX

int **remove_ocvm**
[-coefficient]
[-derate]
[-corners *list_of_corners*]
[*object_list*]

Data Types

list_of_corners list or collection
object_list list

Data Types

object_list list

ARGUMENTS

-coefficient

Removes only the AOCV coefficients. The *-coefficient* and *-derate* options are mutually exclusive.

-derate

Removes only the AOCV/POCV derating factors. The *-coefficient* and *-derate* options are mutually exclusive.

-corners

Specifies the list of corners for which the AOCV/POCV derating factors are removed. If this option is not provided, the corner associated with the current scenario is used.

object_list

If *-coefficient* has been specified, indicates that the AOCV coefficients set on the library cells and cells in the *object_list* are removed. If *-derate* has been specified, indicates that the AOCV/POCV derating factors set on the design, hierarchical cells, and library cells in the *object_list* are removed.

DESCRIPTION

Removes user-specified AOCV coefficients and AOCV/POCV derating factors that were previously set using the **set_aocvm_coefficient** and **read_ocvm** commands respectively. If the command has been specified with no options, all AOCV/POCV data is removed.

To display AOCV/POCV deratings and coefficients, use the **report_ocvm** command.

Multicorner-Multimode Support

By default, this command works on the current corner. To specify a different corner, use the **-corners** option.

EXAMPLES

The following example removes the AOCV coefficients from the library cell named lib/bufA:

```
prompt> remove_ocvm -coefficient lib/bufA
```

The following example removes all AOCV/POCV derating factors that were set using the **read_ocvm** command for the current corner.

```
prompt> remove_ocvm -derate
```

The following example removes all AOCV/POCV derating factors set on the current design for the current corner.

```
prompt> remove_ocvm -derate [current_design]
```

SEE ALSO

read_ocvm(2)
report_ocvm(2)

remove_output_delay

Removes output delay from output ports or pins.

SYNTAX

string **remove_output_delay**
[-clock *clock_name*]
[-clock_fall]
[-level_sensitive]
[-rise]
[-fall]
[-max]
[-min]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
port_pin_list

Data Types

clock_name string
mode_list list
corner_list list
scenario_list list
port_pin_list list

ARGUMENTS

-clock *clock_name*

Specifies the relative clock. Specify {""} as the *clock_name* to remove delays not relative to any clock.

-clock_fall

Removes the delay relative to falling edge of clock. If you specify *clock_name* without **-clock_fall**, the delay relative to rising edge of the clock is removed.

-level_sensitive

Removes level-sensitive output delay.

-rise

Removes rising output delay.

-fall

Removes falling output delay.

-max

Removes maximum output delay.

-min

Removes minimum output delay.

-modes ***mode_list***

Specifies the scenarios that output delays will be removed from. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option cannot be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners ***corner_list***

Specifies the scenarios that output delays will be removed from. If this option is given, all scenarios of the given corners and the current mode will be used. The **-scenarios** option cannot be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios ***scenario_list***

Specifies the scenarios that output delays will be removed from. The **-modes** or **-corners** option cannot be given with this option.

port_pin_list

Specifies a list of ports and pins. Each element in the list is either a collection of ports or pins, or a pattern which matches ports or pins on the current design.

DESCRIPTION

Removes output delay values for objects in the current design. Set output delay with **set_output_delay**. By default, all output delays on each object in *port_pin_list* are removed. To restrict the removed output delay values, use **-clock**, **-clock_fall**, **-min**, **-max**, **-rise**, or **-fall**. To show output delays associated with ports, use **report_ports -output_delay**.

Output delay is managed on a per-scenario basis. For designs with multiple scenarios, you can specify scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command is applied to the current scenario. (It will be an error if there is no current scenario). If the **-scenarios** option is given, the command will be applied to all of the specified scenarios. If the **-modes** option is given, the command will be applied to all scenarios of the specified modes. (It will be an error if none of the specified modes have any scenarios.) If the **-corners** option is given, the command will be applied to all scenarios of the specified corners and the current mode. (It will be an error if none of the specified corners have any scenarios with the current mode.) If both the **-modes** and **-corners** options are given, the command will be applied to all scenarios of the specified corners and the specified modes. (It will be an error there are no such scenarios.) It is an error to give **-scenarios** with **-modes** or **-corners**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example removes all output delay values from all output ports in the current scenario.

```
prompt> remove_output_delay [all_outputs]
```

The following example removes output maximum rise delay values from OUT1, OUT3, and BIDIR12.

```
prompt> remove_output_delay -max -rise { OUT1 OUT3 BIDIR12 }
```

The following example removes all output delays for port OUT1 relative to the falling edge of CLK2.

```
prompt> remove_output_delay -clock CLK2 -clock_fall OUT1
```

The following example removes all output delays not relative to any clock for port OUT2.

```
prompt> remove_output_delay -clock {""} OUT2
```

SEE ALSO

all_outputs(2)
collections(2)
report_ports(2)
set_input_delay(2)
set_output_delay(2)

remove_path_group

Removes path group objects.

SYNTAX

status **remove_path_groups**
-all | *group_list*

Data Types

group_list list

ARGUMENTS

-all

Removes all path groups in the current mode.

group_list

Specifies path groups to remove. Each element in the list is either a collection of path groups or a pattern that matches path group names.

DESCRIPTION

The **remove_path_groups** command removes path groups in the current mode.

The **group_path** and **create_clock** commands create path groups. Path groups affect the cost function for optimization and influence the timing reports. If you remove a path group, any paths in that group are implicitly assigned to the default path group.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example removes all path groups in the current mode.

prompt> **remove_path_groups -all**

The following example removes the path group named BUS1.

prompt> **remove_path_groups BUS1**

SEE ALSO

collections(2)
create_clock(2)
group_path(2)

remove_path_groups

Removes path group objects.

SYNTAX

status **remove_path_groups**
-all | *group_list*

Data Types

group_list list

ARGUMENTS

-all

Removes all path groups in the current mode.

group_list

Specifies path groups to remove. Each element in the list is either a collection of path groups or a pattern that matches path group names.

DESCRIPTION

The **remove_path_groups** command removes path groups in the current mode.

The **group_path** and **create_clock** commands create path groups. Path groups affect the cost function for optimization and influence the timing reports. If you remove a path group, any paths in that group are implicitly assigned to the default path group.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example removes all path groups in the current mode.

prompt> **remove_path_groups -all**

The following example removes the path group named BUS1.

prompt> **remove_path_groups BUS1**

SEE ALSO

collections(2)
create_clock(2)
group_path(2)

remove_path_margin

Removes path margin.

SYNTAX

status **remove_path_margin**
[-comment *comment*]
[-all]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]

Data Types

comment string
mode_list list
corner_list list
scenario_list list

ARGUMENTS

-comment *comment*

Remove path margin with specified comment only.

-all

This option removes path margins with any comment from specified mode, corner or scenario. If no mode, corner or scenario is specified it works in current scenario only.

-modes *mode_list*

Specifies the scenarios to remove path margin. If this option is given, all scenarios of the given modes are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-corners *corner_list*

Specifies the scenarios to remove path margin. If this option is given, all scenarios of the given corners and the current mode is used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-scenarios *scenario_list*

Specifies the scenarios to remove path margin. The **-modes** or **-corners** option must not be given with this option.

DESCRIPTION

This command removes path margin.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

This example removes path margins from scenario s1

prompt> **remove_path_margin -all -scenarios s1**

This example removes all path margin with comment pm2 in mode m1.

prompt> **remove_path_margin -comment pm2 -modes m1**

SEE ALSO

reset_paths(2)
set_path_margin(2)

remove_pg_mask_constraints

Removes the specified PG mask constraints.

SYNTAX

status **remove_pg_mask_constraints**
[-all]
name

Data Types

name string

ARGUMENTS

-all

Removes all predefined PG mask constraints.

name

Specifies the PG mask constraint to remove.

DESCRIPTION

This command removes PG mask constraints.

EXAMPLES

The following example removes the PG mask constraint named rule1.

prompt> **remove_pg_mask_constraints rule1**

The following example removes all PG mask constraints.

prompt> **remove_pg_mask_constraints -all**

SEE ALSO

report_pg_mask_constraints(2)
set_pg_mask_constraint(2)

remove_pg_patterns

Removes the specified power ground patterns.

SYNTAX

status **remove_pg_patterns**
[-all]
pattern_name

Data Types

pattern_name string

ARGUMENTS

-all

Removes all predefined power ground patterns.

pattern_name

Removes the specified power ground pattern.

DESCRIPTION

This command removes power ground patterns such as wire pattern, ring pattern, macro connection pattern, standard cell connection pattern, and composite pattern.

EXAMPLES

The following example removes the power ground pattern named pattern1.

prompt> **remove_pg_patterns pattern1**

The following example removes all power ground patterns.

prompt> **remove_pg_patterns -all**

SEE ALSO

create_pg_wire_pattern(2)
create_pg_ring_pattern(2)
create_pg_macro_conn_pattern(2)
create_pg_std_cell_conn_pattern(2)
create_pg_composite_pattern(2)
create_pg_special_pattern(2)
report_pg_patterns(2)

remove_pg_regions

Removes the specified power ground regions.

SYNTAX

status **remove_pg_regions**
[-force]
[-all]
[*region*]

Data Types

region collection

ARGUMENTS

-force

Ignores locked status of the objects. If this option is not provided and any of the affected objects has locked status, the command fails and generates a Tcl error.

-all

Remove all power ground regions.

region

Removes the specified collection of power ground regions.

DESCRIPTION

This command removes the specified power ground regions.

EXAMPLES

The following example removes the power ground region named region1.

prompt> **remove_pg_regions region1**

The following example removes all power ground regions.

```
prompt> remove_pg_regions -all
```

SEE ALSO

create_pg_region(2)
report_pg_regions(2)

remove_pg_strategies

Removes the specified power ground strategy.

SYNTAX

status **remove_pg_strategies**
[-all]
strategy_name

Data Types

strategy_name string

ARGUMENTS

-all

Removes all power ground strategies. By default, the command removes only the specified power ground strategy.

strategy_name

Removes the specified power ground strategy. You must specify a *strategy_name* or the **-all** option.

DESCRIPTION

This command removes the specified power ground strategy. Use the **-all** option to remove all power ground strategies.

EXAMPLES

The following example removes the top_strategy power ground strategy.

prompt> **remove_pg_strategies top_strategy**

The following example removes all power ground strategies.

prompt> **remove_pg_strategies -all**

SEE ALSO

compile_pg(2)
report_pg_strategies(2)
set_pg_strategy(2)

remove_pg_strategy_via_rules

Removes the specified power ground strategy via rule.

SYNTAX

status **remove_pg_strategy_via_rules**
[-all]
rule_name

Data Types

rule_name string

ARGUMENTS

-all

Removes all power ground strategy via rules. By default, the command removes only the specified power ground strategy via rule.

rule_name

Removes the specified power ground strategy via rule. You must specify a *rule_name* or the **-all** option.

DESCRIPTION

This command removes the specified power ground strategy via rule. Use the **-all** option to remove all power ground strategy via rules.

EXAMPLES

The following example removes the power ground strategy via rule r1.

prompt> **remove_pg_strategy_via_rules r1**

The following example removes all power ground strategy via rules.

prompt> **remove_pg_strategy_via_rules -all**

SEE ALSO

compile_pg(2)
report_pg_strategy_via_rules(2)
set_pg_strategy_via_rule(2)

remove_pg_via_master_rules

Removes the specified power ground via rule.

SYNTAX

status **remove_pg_via_master_rules**
[-all]
rule_name

Data Types

rule_name string

ARGUMENTS

-all

Removes all predefined power ground via rules.

rule_name

Removes the specified power ground via rule.

DESCRIPTION

This command removes the specified power ground via rule.

EXAMPLES

The following example removes power ground via rule rule1.

prompt> **remove_pg_via_master_rules rule1**

The following example removes all power ground via rules.

prompt> **remove_pg_via_master_rules -all**

SEE ALSO

report_pg_via_master_rules(2)
set_pg_via_master_rule(2)

remove_physical_objects

Removes physical shapes of specified objects.

SYNTAX

status **remove_physical_objects**
[*object_list*]
[-remove_disconnected]
[-quiet]

Data Types

object_list collection or selection

ARGUMENTS

object_list

Specifies a collection or selection set containing objects to remove. If this option is not specified, global selection set is used.

-remove_disconnected

Removes any vias that are not longer connected on its upper or lower layer when a new shape is deleted.

Note: this option is not supported in the library manager as this tool does not support the required connectivity analysis functionality.

-quiet

Suppress all messages.

DESCRIPTION

Removes physical shapes of specified objects. It does not remove nets or bundles , but it does remove shapes of specified nets and bundles.

EXAMPLES

The following example removes all bound objects:

```
prompt> change_selection [get_bounds *]
prompt> remove_physical_objects
```

The alternative syntax uses a collection to specify objects.

```
prompt> remove_physical_objects [get_bounds *]
```

SEE ALSO

change_selection(2)
get_selection(2)

remove_physical_rules

Removes abstract physical rules and attributes from target library and all its cells. This command can only remove physical rules and attributes which can only be added by read_physical_rules.

SYNTAX

int **remove_physical_rules**
[-library *lib*]

Data Types

lib string

ARGUMENTS

-library *lib*

Specifies the target library to remove the rules and attributes. This option is only avaiable in ICC2. If not specified, current library will be used.

In library manager, it's always the current worksapce library.

DESCRIPTION

This command removes abstract physical rules/attributes from target library and all its cells.

EXAMPLES

The following example removes all rules from library and cells.

```
prompt> remove_physical_rules
2
```

SEE ALSO

read_physical_rules(2)
write_physical_rules(2)

remove_pin_blockages

Removes pin blockages from the current design.

SYNTAX

status **remove_pin_blockages**
[-force]
[-verbose]
-all | *pin_blockage_list*

Data Types

pin_blockage_list list

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Prints additional messages.

-all

Removes all pin blockages in the current design.

pin_blockage_list

Specifies the pin blockages. The pin_blockage_list can be a collection handle of pin blockages or names of patterns. You can use the *get_pin_blockages* command to specify objects.

RETURN VALUE

The number of removed pin blockages.

DESCRIPTION

This command removes pin blockages from the design.

EXAMPLES

The following example removes pin blockages whose names begin with *PIN_BLOCKAGE*:

prompt> **remove_pin_blockages PIN_BLOCKAGE***

SEE ALSO

create_pin_blockage(2)
get_pin_blockages(2)
report_pin_blockages(2)
add_to_pin_blockage(2)
remove_from_pin_blockage(2)

remove_pin_buses

Removes pin buses from the current design.

SYNTAX

collection **remove_pin_buses**
[-design *design*]
[-force]
[-block *block*]
-all | *pin_bus_list*

Data Types

pin_bus_list string or collection

ARGUMENTS

-design *design*

Specifies the design in which to remove the pin buses. If no design is specified, the pin buses are removed from the current design.

-force

Ignores the locked status of pins of the bus pin. If this option is not specified and any of the affected objects has locked status, the command issues a Tcl error and exits.

-block *block*

Specifies the block where the pin bus is to be removed. If specified, **remove_pin_buses** has the same effect as **edit_module** in which the module is changed and changes will be on all module occurrences.

-all

Removes all pin buses.

pin_bus_list

Specifies a list of pin buses to remove.

RETURN VALUE

This command returns the removed pin bus count, an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

DESCRIPTION

This command removes a list of pin buses from the current design (unless otherwise specified by -design). Individual members of buses remain.

EXAMPLES

The following example removes pin bus named *bus_cell/bus1*

```
prompt> remove_pin_buses bus_cell/bus1
1
```

SEE ALSO

create_pin_bus(2)
get_pin_buses(2)
report_pin_buses(2)

remove_pin_constraints

Remove the specified pin constraint(s).

SYNTAX

int **remove_pin_constraints**
pin_constraints_list

pin_constraints_list collection

ARGUMENTS

pin_constraints_list

A collection of pin constraint objects to be removed.

DESCRIPTION

This command removes a set of pin constraint objects from the database. The command takes a collection of pin constraints as an argument and will error out if the collection does not contain exclusive pin constraints. Upon success the command will return the number of objects removed.

EXAMPLES

The following example removes all pin constraints from the design.

```
prompt> remove_pin_constraints [get_pin_constraints]
1
```

SEE ALSO

place_pins(2)
create_pin_constraint(2)
report_pin_constraints(2)

set_editability(2)

remove_pin_guides

Removes pin guides from the current design.

SYNTAX

status **remove_pin_guides**
[-force]
[-verbose]
-all | *pin_guide_list*

Data Types

pin_guide_list list

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Prints additional messages.

-all

Removes all pin guides in the current design.

pin_guide_list

Specifies the pin guides. The pin_guide_list can be a collection handle of pin guides or names of patterns. You can use the *get_pin_guides* command to specify objects.

RETURN VALUE

The number of removed pin guides.

DESCRIPTION

This command removes pin guides from the design.

EXAMPLES

The following example removes pin guides whose names begin with *PIN_GUIDE*:

prompt> **remove_pin_guides PIN_GUIDE***

SEE ALSO

create_pin_guide(2)
get_pin_guides(2)
report_pin_guides(2)
add_to_pin_guide(2)
remove_from_pin_guide(2)

remove_pin_name_synonym

Removes pin name synonym definitions.

SYNTAX

status **remove_pin_name_synonym**
[-all]
[synonym_list *synonym_list*]

Data Types

synonym_list list

ARGUMENTS

-all

Removes all pin name synonym definitions. You must specify either the -all option or the synonym_list argument. The option and argument are mutually exclusive.

synonym_list

Removes pin name synonym definitions that match one of the pin name synonym strings in the list. You must specify either the -all option or the synonym_list argument. The option and argument are mutually exclusive.

DESCRIPTION

The remove_pin_name_synonym command deletes some or all pin name synonym definitions.

EXAMPLES

The following examples set and remove pin name synonyms:

```
prompt> set_pin_name_synonym SYN_CD CD
1

prompt> set_pin_name_synonym SYN_QN QN
1
```

```
prompt> set_pin_name_synonym \
     -full_name this/is/a/synonym mid1/FJK2SP_at_mid/TI
1

prompt> set_pin_name_synonym \
     -full_name mid1/FJK2SP_at_mid/J mid2/bot2/LSR0P_at_bot/Q
1

prompt> report_pin_name_synonym
****************************************
Report : pin name synonym
Design : top
Version: X-2005.09
Date   : Wed Jun 29 10:29:04 2005
****************************************

Synonym                  Pin Name                 Type
--------------------------------------------------------------------------------
this/is/a/synonym            mid1/FJK2SP_at_mid/TI       full name
mid1/FJK2SP_at_mid/J         mid2/bot2/LSR0P_at_bot/Q    full name
mid1/bot1/LSR0P_at_bot/SYN_R  mid1/bot1/LSR0P_at_bot/R    full name
SYN_QN                   QN                       simple name
SYN_CD                   CD                       simple name
--------------------------------------------------------------------------------
1
```

SEE ALSO

remove_pin_name_synonym(2)
report_pin_name_synonym(2)

remove_pins

Removes pins from a cell.

SYNTAX

remove_count **remove_pins**
[-design *design*]
[-force]
pin_list

pin_list list of pins to remove

ARGUMENTS

-design *design*

Specifies the design in which to remove the pin(s). If no design is specified, the pin(s) are removed from the current design.

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

pin_list

Specifies a list of pins to remove from their cells.

DESCRIPTION

This command removes a list of pins from their cells in the current design (unless otherwise specified by -design).

Any nets connected to removed pins are disconnected from the pins before they are removed. Remaining net connections are not disturbed. Unconnected nets are not removed.

If the cell is an instance of another block, the corresponding ports on that block will also be removed. If the cell is an instance of a multiply instantiated block (MIB) then each cell that instantiates that block will also have its corresponding pins removed.

The **remove_pins** command returns the number of pins from *pin_list* that were removed.

EXAMPLES

The following example removes pin i3 from cell u1.

```
prompt> get_pins u1/*
{u1/i1 u1/i2 u1/i3 u1/o1}
prompt> remove_pins u1/i4
1
prompt> get_pins u1/*
{u1/i1 u1/i2 u1/o1}
```

The following example removes pins from several cells.

```
prompt> remove_pins [get_pins u*/i3]
14
```

The following example removes 3 pins from instance uMID1 of multiply instantiated block MID. The corresponding pins are also removed from the other instance uMID2.

```
prompt> get_pins uMID1/*
{uMID1/i1 uMID1/a1 uMID1/a2 uMID1/a3 uMID1/o1}
prompt> get_pins uMID2/*
{uMID2/i1 uMID2/a1 uMID2/a2 uMID2/a3 uMID2/o1}

prompt> remove_pins uMID1/a*
3

prompt> get_pins uMID1/*
{uMID1/i1 uMID1/o1}
prompt> get_pins uMID2/*
{uMID2/i1 uMID2/o1}
```

SEE ALSO

create_pin(2)
disconnect_net(2)
get_pins(2)
remove_ports(2)

remove_pins_from_virtual_connection

Disconnects pins or ports from the virtual connection.

SYNTAX

status **remove_pins_from_virtual_connection**
-object *virtual_connection*
-pins *pin_port_list*

Data Types

virtual_connection list or collection
pin_port_list list or collection

ARGUMENTS

-object *virtual_connection*

Specifies one virtual connection.

-pins *pin_port_list*

Specifies a list of pins and ports to be disconnected from the virtual connection. The command errors out if any pin is not connected to the specified virtual connection.

DESCRIPTION

This command disconnects pins from the specified virtual connection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples disconnect pins from the specified virtual connection.

prompt> **remove_pins_from_virtual_connection** \

```
        -object [get_virtual_connections -name SNPS_vc_0] \
        -pins {lh1/ff3/Y U2/A U3/A MemData[0]}\

prompt> remove_pins_from_virtual_connection \
        -object [get_virtual_connections -name SNPS_vc_0] \
        -pins [get_pins {U2/A U3/A}]\

prompt> remove_pins_from_virtual_connection \
        -object [get_virtual_connections -name SNPS_vc_0] \
        -pins [get_ports MemData[0]]\
```

SEE ALSO

add_pins_to_virtual_connection(2)
create_virtual_connection(2)
get_attribute(2)
get_virtual_connections(2)
place_eco_cells(2)
remove_virtual_connections(2)
set_attribute(2)

remove_placement_attractions

Removes placement_attractions from the current design.

SYNTAX

int **remove_placement_attractions**
[-force]
[*placement_attraction_list*]

Data Types

placement_attraction_list list

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

placement_attraction_list

Specifies a list of placement_attractions to remove. The list may contain placement_attraction names, patterns, or collections. A collection may be specified by using the **get_placement_attractions** command.

DESCRIPTION

This command removes placement_attraction constraints from the current design. The number of removed placement_attractions are returned.

EXAMPLES

The following example removes placement_attractions named "placement_attraction1" and "placement_attraction2".

```
prompt> remove_placement_attractions { placement_attraction1 placement_attraction2 }
2
```

The following example removes placement_attractions that match the pattern "placement_attraction*".

```
prompt> remove_placement_attractions placement_attraction*
3
```

SEE ALSO

create_placement_attraction(2)
get_placement_attractions(2)
add_to_placement_attraction(2)
remove_from_placement_attraction(2)
report_placement_attractions(2)

remove_placement_blockages

Removes placement blockages.

SYNTAX

status **remove_placement_blockages**
[-force]
[-verbose]
-all | *blockage_list*

Data Types

blockage_list collection

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Show detailed information while removing placement blockages

-all

Remove all placement blockages from the current design. This option is mutually exclusive with the *blockage_list* argument, but you must specify one.

blockage_list

Specifies the list of objects by which this command finds and removes a placement blockage. The list can be a collection of blockages, or a pattern, such as * or PB_* (removes all placement blockages) or PB_7789 (removes one placement blockage whose *object_id* is 7789).

The *blockage_list* and the **-all** option are mutually exclusive.

DESCRIPTION

This command removes all specified placement blockages.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all placement blockages:

prompt> **remove_placement_blockages -all**

The following example uses the *patterns* argument to remove placement blockages:

prompt> **remove_placement_blockages [get_placement_blockages PB_*]**

SEE ALSO

create_placement_blockage(2)
create_routing_guide(2)
get_placement_blockages(2)
get_routing_guides(2)
remove_routing_guides(2)

remove_placement_spacing_rules

Removes placement spacing labels and rules set with *set_placement_spacing_label* and *set_placement_spacing_rule* commands..

SYNTAX

status **remove_placement_spacing_rules**
-label *label_name*
-rule *label_names*
-all

Data Types

label_name	string
label_names	pair of strings

ARGUMENTS

-label *label_name*

Specifies a name for label to be removed. Additionally all rules referencing this label are removed. A built-in label such as "SNPS_BOUNDARY" may not be removed.

-rule *label_names*

Specifies the names of the labels that constitute the rule to be removed. The labels are nore removed.

-all

Specifies that all labels, other than built-in labels, and.rules are to be removed.

Options *-label*, *-rule*, and *-all* are mutually exclusive. One and only one of these options must be specified.

DESCRIPTION

This command removes placement spacing rules that have been set with commands set_placement_spacing_label and set_placement_spacing_rule. The user may optionally remove a single label, rule, or all label and rules. In order to re-instate placement spacing constraints, the user will have to rerun set_placement_spacing_label and set_placement_spacing_rule commands.

EXAMPLES

The following example removes label X

prompt> **remove_placement_spacing_rules -label X**

The following example removes rule between labels X and Y

prompt> **remove_placement_spacing_rules -rule {X Y}**

SEE ALSO

set_placement_spacing_rule(2)
set_placement_spacing_label(2)
report_placement_spacing_rules(2)

remove_pop_up_object_options

Resets options for the **pop_up_objects** command to their default values.

SYNTAX

string **remove_pop_up_object_options**
[-object_type {pg_routing routing_guide blockage pin_guide pin_blockage cells}]

ARGUMENTS

-object_type {pg_routing routing_guide blockage pin_guide pin_blockage cells}

Specifies the types of the objects whose options will be reset. You can specify one or more of the supported object types by using the following keywords: *pg_routing*, *routing_guide and blockage*, *pin_guide*, *pin_blockage* and *cells*. If **-object_type** is not specified, the settings for all object types are reset.

DESCRIPTION

This command resets options for the **pop_up_objects** command to their default values. You can check the default values of each object type with the **report_pop_up_object_options** command.

EXAMPLES

The following example resets the options for *pg_routing*.

prompt> **remove_pop_up_object_options -object_type pg_routing**

SEE ALSO

pop_up_objects(2)
report_pop_up_object_options(2)
set_pop_up_object_options(2)

remove_port_buses

Removes port buses from the current design.

SYNTAX

collection **remove_port_buses**
[-design *design*]
[-force]
[-block *block*]
[-cell *cell*]
[-all]
port_bus_list

Data Types

block string or collection
cell string or collection
port_bus_list string or collection

ARGUMENTS

-design *design*

Specifies the design in which to remove the port bus. If no design is specified, the port bus is removed from the current design.

-block *block*

Specifies the block where the port bus is to be removed. If specified, **remove_port_buses** has the same effect as **edit_module** in which the module is changed and changes will be on all module occurrences.

-cell *cell*

Specifies the cell where the port bus is to be removed. The port bus is removed on the cell's reference module. **remove_port_buses** will first uniquify the reference if needed, then remove the port bus. When not specified, the port bus is removed on the current module.

-force

Ignores the locked status of ports of the bus port. If this option is not specified and any of the affected objects has locked status, the command issues a Tcl error and exits.

-all

Removes all port buses.

port_bus_list

Specifies a list of port buses to remove.

RETURN VALUE

This command returns the removed port bus count, an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

DESCRIPTION

This command removes a list of port buses from the current design (unless otherwise specified by -design). Individual members of buses remain.

EXAMPLES

The following example removes port bus named *bus1*

```
prompt> remove_port_buses bus1
1
```

SEE ALSO

add_to_port_bus(2)
create_port_bus(2)
get_port_buses(2)
remove_from_port_bus(2)
report_port_buses(2)

remove_ports

Removes ports from the current design.

SYNTAX

status_value **remove_ports**
[-design *design*]
[-force]
[-remove_shapes]
port_list

port_list list

ARGUMENTS

-design *design*

Specifies the design in which to remove the port(s). If no design is specified, the port(s) are removed from the current design.

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-remove_shapes

Specifies that pin shapes also need to be removed.

port_list

Specifies a list of ports to remove from the current design.

DESCRIPTION

This command removes a list of ports from the current design (unless otherwise specified by -design).

EXAMPLES

The following example removes all ports.

```
prompt> get_ports *
{"VDD", "VSS", "data2", "data1"}
```

The following example removes the port named *data2*.

```
prompt> remove_ports data2
1
```

The following example removes the ports that start with *data*, specified by the **get_ports** command.

```
prompt> set a [get_ports data*]
{"data1"}
prompt> remove_ports $a
1
```

The following example removes the port named *data3* along with pin shapes.

```
prompt> remove_ports -remove_shapes data3
1
```

SEE ALSO

create_port(2)
get_ports(2)
remove_pins(2)

remove_post_route_filler

Removes unfixed filler cells inserted with the **create_stdcell_fillers -rules {post_route_auto_delete}** command.

SYNTAX

status **remove_post_route_filler**
[-skip_filler_type_check]
[-hierarchical]

Data Types

No data types

ARGUMENTS

-skip_filler_type_check

If this option is specified, the command checks cells of other cell types. By default, the command only checks and removes cells that are physical_only or are filler cell types.

-hierarchical

If this option specified, the command checks cells in sub-blocks and at the top level. By default, command only checks and removes cells only in the current block.

DESCRIPTION

This command deletes filler cells in the route_opt flow. It deletes all unfixed filler cells inserted with the **create_stdcell_fillers -rules {post_route_auto_delete}** command.

EXAMPLES

The following example first inserts filler cells with post_route_auto_delete option, then removes them all with the **remove_post_route_filler** command.

```
prompt> create_stdcell_fillers \
  -lib_cells {mylib/FILL_4X mylib/FILL_2X mylib/FILL_1X}
```

```
prompt> remove_post_route_filler
```

SEE ALSO

create_stdcell_fillers(2)

remove_power_io_constraints

Removes power constraints from the specified I/O guides.

SYNTAX

int **remove_power_io_constraints**
[-io_guide_list *io_guides*]

Data Types

io_guides list

ARGUMENTS

-io_guide_list

Specifies the names or collection of target I/O guides. If this option is not used, all I/O guides are processed.

DESCRIPTION

This command removes power constraints from I/O guides.

EXAMPLES

The following example removes power constraints from an I/O guides io_guide1 and io_guide2.

prompt> **remove_power_io_constraints -io_guide_list {io_guide1 io_guide2}**

The following example removes power constraints from all I/O guides.

prompt> **remove_power_io_constraints**

SEE ALSO

place_io(2)
report_power_io_constraints(2)
set_power_io_constraints(2)
set_signal_io_constraints(2)

remove_pr_rules

Removes cell row spacing rules (pr_rules) from the current technology object.

SYNTAX

integer **remove_pr_rules**
[-all]
[*pr_rule_list*]

Data Types

pr_rule_list collection

ARGUMENTS

-all

Removes all cell row spacing rules.

The **-all** option and *pr_rule_list* argument are mutually exclusive; you must specify one.

pr_rule_list

Specifies the cell row spacing rules to remove. You can specify the rules by name or as a collection.

The **-all** option and *pr_rule_list* argument are mutually exclusive; you must specify one.

DESCRIPTION

This command removes cell row spacing rules from the current technology object.

This command returns the number of rules removed. If there is a command syntax error, it raises a TCL_ERROR.

EXAMPLES

The following example removes the cell row spacing rule named PR_RULE_0.

prompt> **remove_pr_rules PR_RULE_0**

1

SEE ALSO

create_pr_rule(2)
get_pr_rules(2)
report_pr_rules(2)

remove_programmable_spare_cell_mapping_rule

This command remove rules for programmable spare cells mapping.

SYNTAX

status **remove_programmable_spare_cell_mapping_rule**
-psc_type_id *id_list* | -all

Data Types

id_list list

ARGUMENTS

-psc_type_id *id_list*

Specify the list of psc_type_id of which the mapping rules will be removed. This option is mutually exclusive with option *-all*.

-all *id_list*

Remove all mapping rules for all psc types. This option is mutually exclusive with option *-psc_type_id*.

DESCRIPTION

This command remove rules for programmable spare cells mapping. Users can specify the list of psc_type_id to remove mapping rules or remove all mapping rules for all types.

After removing mapping rules for a psc_type_id, the psc mapping of the command **place_freeze_silicon** does not honor any mapping rule for this psc_type_id.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example remove the rule for psc_type_id 1 and 2.

prompt> **remove_programmable_spare_cell_mapping_rule -psc_type_id {1 2}**\

The following example remove all rules.

prompt> **remove_programmable_spare_cell_mapping_rule -all**\

SEE ALSO

set_programmable_spare_cell_mapping_rule(2)
report_programmable_spare_cell_mapping_rule(2)
place_freeze_silicon(2)
map_freeze_silicon(2)

remove_propagated_clock

Removes a propagated clock specification.

SYNTAX

string **remove_propagated_clocks** *object_list*

list *object_list*

ARGUMENTS

object_list

Lists clocks, ports, or pins.

DESCRIPTION

Removes propagated clock specification from clocks, ports, or pins in the current design. The **set_propagated_clock** command specifies that delays be propagated through the clock network to determine latency at register clock pins. If this is not specified, ideal clocking is assumed. Ideal clocking means clock networks have a user specified latency (set by the **set_clock_latency** command) or zero latency, by default. Propagated clock latency is normally used for post layout, after final clock tree generation.

Ideal clock latency provides an estimate of the clock tree for prelayout. You can also use **set_clock_latency** to specify an ideal latency, which overrides the propagated clock specification for an object.

For information on propagated clock attributes on clocks, see the **report_clock** command man page.

Multicorner-Multimode Support

EXAMPLES

The following example removes propagated clock specifications from all clocks in the design.

```
prompt> remove_propagated_clocks [all_clocks]
```

SEE ALSO

report_clock(2)
set_clock_latency(2)
set_propagated_clock(2)

remove_propagated_clocks

Removes a propagated clock specification.

SYNTAX

string **remove_propagated_clocks** *object_list*

list *object_list*

ARGUMENTS

object_list

Lists clocks, ports, or pins.

DESCRIPTION

Removes propagated clock specification from clocks, ports, or pins in the current design. The **set_propagated_clock** command specifies that delays be propagated through the clock network to determine latency at register clock pins. If this is not specified, ideal clocking is assumed. Ideal clocking means clock networks have a user specified latency (set by the **set_clock_latency** command) or zero latency, by default. Propagated clock latency is normally used for post layout, after final clock tree generation.

Ideal clock latency provides an estimate of the clock tree for prelayout. You can also use **set_clock_latency** to specify an ideal latency, which overrides the propagated clock specification for an object.

For information on propagated clock attributes on clocks, see the **report_clock** command man page.

Multicorner-Multimode Support

EXAMPLES

The following example removes propagated clock specifications from all clocks in the design.

```
prompt> remove_propagated_clocks [all_clocks]
```

SEE ALSO

report_clock(2)
set_clock_latency(2)
set_propagated_clock(2)

remove_purposes

Removes technology purposes from the technology data.

SYNTAX

collection **remove_purposes**
purpose_list | *-all*

Data Types

purpose_list collection

ARGUMENTS

purpose_list

Specifies the collection of purposes to be removed from the technology data.

-all

Removes all purposes. The **-all** and purpose_list options are mutually exclusive.

DESCRIPTION

This command removes the specified technology purposes from the library's technology data. The **remove_purposes** command will not remove the default technology purpose, "drawing". If you try to remove the "drawing" purpose, the tool will issue a warning message.

RETURN VALUE

This command returns the removed technology purpose count, or an empty string if the command fails. If a command syntax error exists, the command returns a TCL_ERROR.

EXAMPLES

The following example removes a purpose named 'test_purpose' from the technology data of the current library.

prompt> **remove_purposes test_purpose**

{1}

SEE ALSO

create_purpose(2)
get_purposes(2)

remove_push_down_object_options

Resets options for the **push_down_objects** command to their defaults.

SYNTAX

string **remove_push_down_object_options**
[-object_type *types*

Data Types

type list

ARGUMENTS

-object_type *types*

Specifies the types of objects whose options are reset. Specify one or more of the following object types: **pg_routing**, **signal_routing**, **routing_guide**, **routing_corridor**, **blockage**, **pin_guide**, **pin_blockage**, **cells**, and **charging_station**.

If you do not specify this option, the command resets the settings for all object types.

DESCRIPTION

This command resets options for the **push_down_objects** command to their defaults. You can report the current push down settings with the **report_push_down_object_options** command.

EXAMPLES

The following example resets the options for *pg_routing*.

prompt> **remove_push_down_object_options -object_type pg_routing**

SEE ALSO

push_down_objects(2)
report_push_down_object_options(2)
set_push_down_object_options(2)

remove_qor_snapshot

Removes an existing QoR snapshot from the location specified by the application option **time.snapshot_storage_location**.

SYNTAX

void **remove_qor_snapshot**
[-name *snapshot_name* | -all]

Data Types

snapshot_name string

ARGUMENTS

-name *snapshot_name*

Remove QoR snapshot named **snapshot_name** for this design, from the location designated by **time.snapshot_storage_location**. The **-name**, and **-all** options are mutually exclusive; you must choose only one.

-all

Remove all QoR snapshots for this design, from the location designated by **time.snapshot_storage_location**. The **-name**, and **-all** options are mutually exclusive; you must choose only one.

DESCRIPTION

This command removes one or more QoR snapshots.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

In the following example, the **remove_qor_snapshot** command removes the QoR snapshot named "preroute" that is saved under the design/snapshot directory.

```
prompt> remove_qor_snapshot -name preroute
```

SEE ALSO

create_qor_snapshot(2)
report_qor_snapshot(2)
query_qor_snapshot(2)
time.snapshot_storage_location(3)

remove_redundant_shapes

Removes dangling wires and vias on user-specified nets.

SYNTAX

status **remove_redundant_shapes**
[-nets {*collection_of_nets*}]
[-route_types {*list_of_route_types*}]
[-layers {*collection_of_layers*}]
[-initial_drc_from_input true | false]
[-remove_loop_shapes true | false]
[-remove_floating_shapes true | false]
[-remove_dangling_shapes true | false]
[-report_changed_nets true | false]

Data Types

{*collection_of_nets*} collection
{*list_of_route_types*} list
{*collection_of_layers*} collection

ARGUMENTS

-nets {*collection_of_nets*}

Specifies the nets from which to remove dangling wires and vias.

If you do not specify this option, dangling wires and vias are removed from all nets.

-route_types {*list_of_route_types*}

Specifies which route types to remove. The valid values are fixed_route and nonfixed_route.

If you do not specify this option, wires with both fixed_route and nonfixed_route route types are removed.

-layers {*collection_of_layers*}

Specifies the layers from which to remove dangling wires or vias. Metal and cut layer names in technology file are supported.

If you do not specify this option, dangling wires and vias are removed from all layers.

-initial_drc_from_input true | false

Specifies whether to run the **check_routes** command before removing the dangling wires and vias.

By default (true), the tool uses the DRC information stored in the CEL view. If you specify **false**, the tool runs the **check_routes**

command.

-remove_loop_shapes true | false

Specifies whether loops are removed.

The default is false.

-remove_floating_shapes true | false

Specifies whether floating shapes are removed.

The default is true.

-remove_dangling_shapes true | false

Specifies whether dangling shapes are removed.

The default is true.

-report_changed_nets true | false

Specifies whether trimmed nets are reported.

The default is false.

DESCRIPTION

The **remove_redundant_shapes** command remove dangling wires and vias on the specified nets.

By default, this command runs the **check_routes** command and then removes the dangling and floating net shapes from all nets in the design, based on the results. When removing dangling net shapes, the tool does not change topologies or connections and does not touch terminals. In addition, no changes are made to open nets or nets with DRC violations.

EXAMPLES

The following example removes the dangling wires and vias from the n1 net.

prompt> **remove_redundant_shapes -nets n1**

The following example removes only those dangling wires and vias that have a route type of nonfixed_route.

prompt> **remove_redundant_shapes -route_types {nonfixed_route}**

The following example removes the dangling wires and vias from the Metal2, Via23, and Metal3 layers.

prompt> **remove_redundant_shapes -layers {METAL2 VIA23 METAL3}**

The following example does not run the **check_routes** command before removing the dangling wires and vias. Instead, it uses the DRC information stored in the CEL view.

prompt> **remove_redundant_shapes -initial_drc_from_input true**

The following example removes loops in addition to dangling wires and vias.

prompt> **remove_redundant_shapes -remove_loop_shapes true**

The following example removes only dangling wires and vias and not floating wires and vias.

prompt> **remove_redundant_shapes -remove_floating_shapes false**

The following example removes only floating wires and vias and not dangling wires and vias.

prompt> **remove_redundant_shapes -remove_dangling_shapes false**

SEE ALSO

check_routes(2)

remove_reference_only_related_supply

Removes the reference-only supply net or supply set that is used as related supply of boundary ports.

SYNTAX

status **remove_reference_only_related_supply**
supply

Data Types

supply string

ARGUMENTS

supply

Specifies the reference-only supply net or supply set to be removed.

DESCRIPTION

The **remove_reference_only_related_supply** command removes the reference-only supply net or supply set that is used as related supply of boundary ports in the current design. It is an error if the specified supply is not reference-only or used in other UPF constraints, except for setting related supply net, driver/receiver supply and attribute that declaring the supply as reference-only.

EXAMPLES

The following example shows a typical use of the *remove_reference_only_related_supply* command:

```
prompt> remove_reference_only_related_supply vddref
1
```

SEE ALSO

create_supply_net(2)

create_supply_set(2)
set_related_supply_net(2)
set_port_attributes(2)

remove_regular_multisource_clock_tree_options

It removes options from command set_regular_multisource_cloc_tree_options for performing auto tap synthesis and global clock tree synthesis for regular multisource clock trees.

SYNTAX

status **remove_regular_multisource_clock_tree_options**

DESCRIPTION

The **remove_regular_multisource_clock_tree_options** command removes options set through the corresponding set option command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example sets the auto tap synthesis options, removes and reports the options.

```
prompt> set_regular_multisource_clock_tree_options -clock clk \
  -net clk -tap_lib_cells [get_lib_cells */CKBUF*] \
  -htree_lib_cells [get_lib_cells */CKBUF*] \
  -topology htree_only -tap_boxes {4 4}
prompt> remove_regular_multisource_clock_tree_options
prompt> report_regular_multisource_clock_tree_options
```

SEE ALSO

synthesize_regular_multisouce_clock_trees(2)
set_regular_multisouce_clock_tree_options(2)
report_regular_multisouce_clock_tree_options(2)

remove_repeater_group_constraints

Removes constraints that are assigned to groups of repeaters.

SYNTAX

status **remove_repeater_group_constraints**
-type *type_list*

Data Types

type_list list

ARGUMENTS

-type *type_list*

Specifies the type of repeater group constraints. The valid values are, "placement", "insertion" and "all". When you specify "placement", the constraints for repeater groups placement are removed. Afterwards, when you run the place_group_repeaters command, this command uses the default settings or the command option settings. When you specify "insertion", the constraints for repeater groups insertion are removed. Afterwards, when you run the add_group_repeaters command, this command uses the default settings or the command option settings. When you specify "all", all the constraints for repeater groups placement and insertion are removed. Afterwards, when you run the place_group_repeaters command or the add_group_repeaters command, these commands use the default settings or the command option settings.

DESCRIPTION

This command removes constraints that are assigned to groups of repeaters.

EXAMPLES

The following examples show the usages of command.

prompt> **set_repeater_group_constraints -type insertion -horizontal_repeater_spacing 7 -layer_cutting_distance {{M1 2.12 13.2} {M2 0.5 0.78} {M3 3.01 3.78}}**
prompt> **set_repeater_group_constraints -type placement -horizontal_repeater_spacing 10**
prompt> **remove_repeater_group_constraints -type insertion**
prompt> **report_repeater_group_constraints**

Report Format:

```
 Type:  Placement
-------------------------------------------------------
 Horizontal repeater spacing: 10
 Vertical repeater spacing: -
 Layer cutting distance: { }

 -------------------------------------------------------
```

SEE ALSO

set_repeater_group_constraints(2)
report_repeater_group_constraints(2)

remove_route_aware_estimation

Prepares the block for final routing when preroute optimization has been run with global-route-based RC estimation.

SYNTAX

status **remove_route_aware_estimation**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command removes all global-route-based estimation from a block to prepare it for final routing with the **route_global** or **route_auto** commands.

The global-route-based estimation information is set on nets in the block when the **opt.common.use_route_aware_estimation** application option is **true** during preroute optimization.

SEE ALSO

clock_opt(2)
place_opt(2)
refine_opt(2)
route_auto(2)
route_global(2)
opt.common.use_route_aware_estimation(3)

remove_routes

Removes net shapes for the specified nets.

SYNTAX

status **remove_routes**
[-nets *nets*]
[-net_types *list_of_net_types*]
[-detail_route]
[-global_route]
[-user_route]
[-shield_route]
[-ring]
[-stripe]
[-lib_cell_pin_connect]
[-macro_pin_connect]
[-follow_pin]
[-core_wire]
[-zero_skew]
[-area_fill]
[-opc]
[-rdl]
[-keep_pg_pins_at_boundary]
[-keep_frozen_nets]

Data Types

nets list or collection
list_of_net_types list

ARGUMENTS

-nets *nets*

Specifies the nets for which to remove net shapes.

-net_types *list_of_net_types*

Specifies the types of nets for which to remove net shapes. Specify one or more of the following values: **analog_ground**, **analog_power**, **analog_signal**, **clock**, **power**, **ground**, **signal**,**tie_high**, and **tie_low**.

If a net specified in the **-nets** option does not have one of the net types specified in this option, the net is ignored. If the design does not contain any net shapes with the specified net types and shape use, the command does not remove any net shapes.

-detail_route

Removes net shapes with a shape use of detail route.

-global_route

Removes net shapes with a shape use of global route.

-user_route

Removes net shapes with a shape use of user route.

-shield_route

Removes net shapes with a shape use of shield route.

-ring

Removes net shapes with a shape use of ring.

-stripe

Removes net shapes with a shape use of stripe.

-lib_cell_pin_connect

Removes net shapes with a shape use of lib cell pin connect.

-macro_pin_connect

Removes net shapes with a shape use of macro pin connect.

-follow_pin

Removes net shapes with a shape use of follow pin.

-core_wire

Removes net shapes with a shape use of core wire.

-zero_skew

Removes net shapes with a shape use of zero skew.

-area_fill

Removes net shapes with a shape use of area fill.

-opc

Removes net shapes with a shape use of opc.

-rdl

Removes net shapes with a shape use of rdl.

-keep_pg_pins_at_boundary

Keeps the PG boundary pins generated by PG routing of standard cells.

By default, pins covered by PG net shapes at the cell boundary are removed with the PG net shapes.

If you specify this option, the pins are preserved and cannot be removed by subsequent **remove_routes** runs, because they are no longer covered by PG routes.

-keep_frozen_nets

Keeps the net shapes of frozen nets, which have a **physical_status** attribute value of **locked**.

DESCRIPTION

This command deletes the net shapes that meet the specified criteria. You can specify the net names, net types, and shape uses.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the net shapes for all detailed routed signal nets.

```
prompt> remove_routes -net_types signal -detail_route
```

remove_routing_blockages

Removes the specified routing blockages.

SYNTAX

status **remove_routing_blockages**
[-force]
[-verbose]
-all | *blockage_list*

Data Types

patterns list

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating Tcl error.

-verbose

Show detailed information while removing routing blockages

-all

Remove all routing blockages from the current design. This option is mutually exclusive with the *blockage_list* argument, but you must specify one.

blockage_list

Specifies the routing blockages. You can specify a collection or the patterns by using the following formats:

- A collection of routing blockages
- An asterisk (*), which indicates all routing blockages
- A string, for example, RB_1234 matches the routing blockage named RB_1234

DESCRIPTION

This command removes the specified routing blockages.

EXAMPLES

The following example removes all routing blockages within the rectangle with the lower-left corner at {2 2} and the upper-right corner at {25 25}.

```
prompt> remove_routing_blockages \
  [get_objects_by_location -classes routing_blockage -within {{2 2} {25 25}}]
1
```

The following example removes all metal routing blockages.

```
prompt> remove_routing_blockages [get_routing_blockages -type metal]
1
```

SEE ALSO

create_routing_blockage(2)
get_routing_blockages(2)

remove_routing_corridor_shapes

Removes routing corridor shapes from the current design.

SYNTAX

int **remove_routing_corridor_shapes**
[-force]
[-verbose]
[-all]
[*routing_corridor_shape_list*]

Data Types

routing_corridor_shape_list list

ARGUMENTS

-force

Ignores the locked status of the objects. If this option is not provided and any of the specified objects has a locked status, the command generates a Tcl error and exits without removing the shape.

-verbose

Displays additional debugging information.

-all

Removes all routing corridors from the current design.

routing_corridor_shape_list

Specifies a list of corridor shapes to remove. The list can contain routing corridor shape collections specified by using the **get_routing_corridor_shapes** command.

DESCRIPTION

This command removes corridor shapes from the current design. If all corridor shapes of a particular routing corridor are removed, the routing corridor is also removed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes one corridor shape named CORRIDOR_SHAPE_1.

```
prompt> remove_routing_corridor_shapes CORRIDOR_SHAPE_1
1
```

The following example removes all routing corridor shapes from the routing corridor named RC_1.

```
prompt> remove_routing_corridor_shapes \
    [get_routing_corridor_shapes -of_objects {RC_1}]
2
```

SEE ALSO

add_to_routing_corridor(2)
create_routing_corridor(2)
create_routing_corridor_shape(2)
remove_from_routing_corridor(2)
remove_routing_corridors(2)
report_routing_corridors(2)

remove_routing_corridors

Removes routing corridors from the current design.

SYNTAX

int **remove_routing_corridors**
[-force]
[-verbose]
[-all]
[*routing_corridor_list*]

Data Types

routing_corridor_list list

ARGUMENTS

-force

Ignores the locked status of the objects. If this option is not provided and any of the specified objects has locked status, the command issues a Tcl error and exits without removing the routing corridor.

-verbose

Displays additional debugging information.

-all

Removes all routing corridors from the current design.

routing_corridor_list

Specifies a list of routing corridors to remove. The list can contain routing corridor names, patterns, or collections. A collection can be specified by using the **get_routing_corridors** command.

The *routing_corridor_list* and **-all** options are mutually exclusive; you must specify only one.

DESCRIPTION

This command removes the specified routing corridors from the current design. The number of removed routing corridors is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the routing corridors whose names start with "clock_corridor".

```
prompt> remove_routing_corridors [get_routing_corridors clock_corridor*]
2
```

The following example removes all corridors from the current design.

```
prompt> remove_routing_corridors -all
7
```

SEE ALSO

add_to_routing_corridor(2)
create_routing_corridor(2)
create_routing_corridor_shape(2)
remove_from_routing_corridor(2)
report_routing_corridors(2)
report_routing_corridors(2)

remove_routing_guides

Removes routing guides.

SYNTAX

int **remove_routing_guides**
[-force]
[-verbose]
-all | *patterns*

Data Types

routing_guide_name string
patterns collection

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Show detailed information while removing routing guides

-all

Remove all routing guides from the current design. This option is mutually exclusive with the *patterns* argument, but you must specify one.

patterns

Specifies the routing guides to remove. This argument is mutually exclusive with the **-all** option, but you must specify one.

DESCRIPTION

The **remove_routing_guides** command removes all specified routing guides.

EXAMPLES

The following example removes the routing guides specified by the **get_routing_guides** command:

```
prompt> remove_routing_guides [get_routing_guides -within {{2 2} {25 25}}]
1
```

The following example removes all routing guides:

```
prompt> remove_routing_guides -all
1
```

SEE ALSO

create_routing_guide(2)
get_routing_guides(2)

remove_routing_rules

Removes non-default routing rules from the design.

SYNTAX

int **remove_routing_rules**
[-verbose]
[-all]
[*routing_rule_list*]

Data Types

routing_rule_list list

ARGUMENTS

-verbose

Displays verbose deletion information.

-all

Removes all non-default routing rules.

routing_rule_list

Removes the non-default routing rules with the specified names.

DESCRIPTION

This command removes the non-default routing rules defined in the design. If you use the **-all** option, all non-default routing rules in the design are removed. If you specify a list, the non-default routing rules in the list are removed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the non-default routing rule named *wire_rule1*.

prompt> **remove_routing_rules wire_rule1**

SEE ALSO

create_routing_rule(2)
report_routing_rules(2)
set_routing_rule(2)

remove_rp_group_options

Remove relative placement constraints from the specified relative placement groups.

SYNTAX

int **remove_rp_group_options**
rp_group_list
[-group_orientation]
[-anchor_corner]
[-rp_only_keepout_margin]
[-allow_non_rp_cells]
[-allow_non_rp_cells_on_blockages]

Data Types

rp_group_list list or collection

ARGUMENTS

rp_group_list

Specifies the list of names or collection of a relative placement groups, from which the given relative placement constraints will be removed.

-group_orientation

Removes orientation constraint of a relative placement group.

-anchor_corner

Removes anchor_corner, x_offset, y_offset, anchor_row, anchor_column constraints of a relative placement group.

-rp_only_keepout_margin

Removes rp_only_keepout_margin from a relative placement group.

-allow_non_rp_cells

Removes allow_non_rp_cells from a relative placement group.

-allow_non_rp_cells_on_blockages

Removes allow_non_rp_cells_on_blockages from a relative placement group.

DESCRIPTION

The **remove_rp_group_options** command removes the placement constraints of relative placement groups.

EXAMPLES

The following example uses the **remove_rp_group_options** command to remove the constraint for a relative placement group.

```
prompt> remove_rp_group_options grp_ripple -group_orientation
1
```

SEE ALSO

set_rp_group_options(2)
create_rp_group(2)
add_to_rp_group(2)
remove_from_rp_group(2)
get_rp_groups(2)
write_rp_groups(2)
remove_rp_groups(2)

remove_rp_groups

Removes relative placement groups from the current design.

SYNTAX

int **remove_rp_groups**
rp_group_list [-hierarchical] | -all

Data Types

rp_group_list list or collection

ARGUMENTS

-all

Removes all relative placement groups from the current block. Please note that relative placement groups of child blocks are not removed. This option is mutually exclusive with *rp_group_list*.

rp_group_list

Specifies a list of relative placement groups to remove. The list may contain relative placement group names, patterns, or collections. A collection may be specified using the **get_rp_groups** command. This option is mutually exclusive with *-all*.

-hierarchical

Specifies that hierarchical relative placement groups at all levels should be removed too, along with specified groups. By default, hierarchical relative placement groups are not removed.

DESCRIPTION

This command removes relative placement groups from the current design. The number of removed groups are returned.

EXAMPLES

The following example removes relative placement group named "top_rp1" and "top_rp2".

```
prompt> remove_rp_groups { top_rp1 top_rp2 }
2
```

The following example removes relative placement groups that match the pattern "rp**".

```
prompt> remove_rp_groups rp*
3
```

The following example removes all relative placement group in the design.

```
prompt> remove_rp_groups -all
5
```

SEE ALSO

create_rp_group(2)
get_rp_groups(2)
add_to_rp_group(2)
remove_from_rp_group(2)
write_rp_groups(2)

remove_sadp_track_rule

Removes a track rule.

SYNTAX

int **remove_sadp_track_rule**
rules_names | -all

Data Types

rules_names string

ARGUMENTS

rule_names

Specifies the list of SADP track rules to remove. You must specify one or more SADP track rule names or use the **-all** option.

-all

Removes all SADP track rules. You must specify the **-all** option or specify SADP track names.

DESCRIPTION

This command removes the specified self-aligned double patterning (SADP) track rule created with the **create_sadp_track_rule** command.

EXAMPLES

The following examples removes all SADP track rules from the current library.

```
prompt> remove_sadp_track_rule -all
Track rule "tr1" is removed
Track rule "tr2" is removed
Track rule "tr3" is removed
3
```

The following example removes SADP track rules tr1 and tr2 from the current library.

```
prompt> remove_sadp_track_rule {tr1 tr2}
Track rule "tr1" is removed
Track rule "tr2" is removed
2
```

SEE ALSO

check_sadp_tracks(2)
create_sadp_track_rule(2)
generate_sadp_tracks(2)
report_sadp_track_rule(2)

remove_safety_register_groups

Removes safety register groups from the current design.

SYNTAX

status **remove_safety_register_groups**
[*objects*]
[-all]

Data Types

objects collection

ARGUMENTS

objects

Specifies the group objects or names which need to be removed

-all

Removes all the safety_register_group objects defined in the current design.

DESCRIPTION

Removes (deletes) the specified group objects from the current design.

EXAMPLES

The following example creates a collection of safety_register_group objects and then deletes them.

prompt> **remove_safety_register_groups [get_safety_register_groups group1*]**

SEE ALSO

create_safety_register_group(2)
report_safety_register_groups(2)
write_safety_register_script(2)

remove_safety_register_rules

Removes safety register rules from the current design.

SYNTAX

status **remove_safety_register_rules**
[*objects*]
[-all]
[-from *registers*]

Data Types

objects	collection
registers	collection

ARGUMENTS

objects

Specifies the rule objects or names which need to be removed

-all

Removes all the safety_register_rule objects defined in the current design.

-from *registers*

Dissociates the safety_register_rule objects from the specified registers. In other words, it deletes the rule-register association.

DESCRIPTION

Removes the specified rule objects from the current design. If the specified rules are being referenced by any safety_register_group, the command will error out with an appropriate warning.

EXAMPLES

The following example creates a collection of safety_register_rule objects and then deletes them.

prompt> **remove_safety_register_rules [get_safety_register_rules rule1*]**

SEE ALSO

create_safety_register_rule(2)
report_safety_register_rules(2)
write_safety_register_script(2)

remove_scaling_lib_group

Remove a scaling group created with **define_scaling_lib_group**

SYNTAX

string **remove_scaling_lib_group**
group_name | -reflibs libs

Data Types

string *group_name*
collection *libs*

ARGUMENTS

group_name

The name of a scaling group already created. Group names are unique within the system. This argument is exclusive with the *-reflibs* argument, and one or the other is required.

-reflibs *libs*

This is a list of one or more reference libraries in which to remove **ALL** scaling groups.

DESCRIPTION

The **remove_scaling_lib_group** command is provided to allow removal of unneeded or incorrectly created scaling groups with **define_scaling_lib_group**. You can remove a specific group by name, or you can remove all groups in one or more libraries. If all groups are removed from a library, then all traces of them are removed. If a single group is removed by name, and it is not the last group created, then **report_lib** will still show an inactive pane for that group.

EXAMPLES

This example shows the removal of a scaling group.

```
prompt> remove_scaling_lib_group -reflib stdcell.ndm
Information: Removed scaling library group G1 in lib stdcell.ndm (SG-010)
```

SEE ALSO

define_scaling_lib_group(2)

remove_scan_chains

Removes scan chains from the current design.

SYNTAX

int **remove_scan_chains**
[-verbose]
[-all | *scan_chains*]

Data Types

scan_chains collection

ARGUMENTS

-verbose

Display verbose deletion information.

-all

Removes all scan chains in the current design.

scan_chains

Specifies a list of scan chains to remove. The list may contain scan chain collections, specified using the **get_scan_chains** command.

DESCRIPTION

This command removes all specified scan chains.

The number of removed scan chains is returned.

EXAMPLES

The following example removes scan chains:

```
prompt> set a [get_scan_chains SCAN*]
{"SCAN1", "SCAN2", "SCAN3"}

prompt> remove_scan_chains $a
3
```

The following example removes scan chains specified by a pattern list:

```
prompt> remove_scan_chains SCAN* -verbose
Removed scan chain SCAN1
Removed scan chain SCAN2
Removed scan chain SCAN3
3
```

SEE ALSO

create_scan_chain(2)
get_scan_chains(2)

remove_scan_def

Removes any SCANDEF scan chain data in current design.

SYNTAX

status **remove_scan_def**

DESCRIPTION

This command removes any SCANDEF scan chain data in current design. This scan chain data has been previously initialized by using the **read_def** command to read a scan Design Exchange Format (DEF) file, also referred to as a SCANDEF file. The **remove_scan_def** command does not affect the design netlist.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

read_def(2)
check_scan_chain(2)
report_scan_chains(2)

remove_scan_rp_group

Removes a DFT Scan relative placement group setup.

SYNTAX

status **remove_scan_rp_group**
rp_group_list
[-all]

Data Types

rp_group_list list or collection

ARGUMENTS

rp_group_list

Specifies the name, a list or a collection of the relative placement groups to be removed from the Scan RP group setup. All groups provided, or their names, must be valid relative placement groups, defined by command **create_rp_group**.

This argument is incompatible with the *-all* argument, and it must be used in an either-or fashion.

-all

Specifies that all Scan RP group setup for all relative placement groups must be removed.

This argument is incompatible with the *rp_group_list* argument, and it must be used in an either-or fashion.

DESCRIPTION

The **remove_scan_rp_group** command removes the setup defined previously by the command **set_scan_rp_group**.

The expected result of the command will be observed by the **report_scan_rp_group** command, and the RP group Scan cells will be used by DFT Scan Synthesis as normal individual Scan cells, allowing DFT Scan Synthesis architecting to put them freely in different Scan chains and/or in arbitrary order, not following any specific order or relationship between other RP group Scan cells but the needs of the overall DFT setup.

Using this command after the design was DFT inserted during commands **compile_fusion** or **insert_dft**, and therefore the Scan chains were already stitched, will not change the original stitching outcome.

EXAMPLES

The following example removes a Scan relative placement setup for RP group named "top_rp".

prompt> **remove_scan_rp_group top_rp**

The following example removes Scan relative placement setup for all RP group that match the pattern "*bottom_rp*".

prompt> **remove_scan_rp_group [get_rp_groups *bottom_rp*]**

The following example removes Scan relative placement setup for all RP group of the design.

prompt> **remove_scan_rp_group -all**

SEE ALSO

create_rp_group(2)
get_rp_groups(2)
write_scan_def(2)
compile_fusion(2)
insert_dft(2)
set_scan_rp_group(2)
report_scan_rp_group(2)

remove_scan_skew_group

Removes scan skew groups defined by the **set_scan_skew_group** command.

SYNTAX

status **remove_scan_skew_group**
scan_skew_group_list

Data Types

scan_skew_group_list list

ARGUMENTS

scan_skew_group_list

Specifies the list of scan skew groups to remove. Wildcards and collections are not supported.

DESCRIPTION

This command removes the specified scan skew groups, which were previously defined by the set_scan_skew_group command.

EXAMPLES

Consider a case where three scan skew groups are initially defined:

```
prompt> report_scan_skew_group

****************************************
Report : Scan Skew Group
Design : top
Version: O-2018.06-SP5-VAL
Date   : Thu Aug 15 00:00:00 2019
****************************************

----------------------------------------
TEST MODE: all_dft
```

```
VIEW    : Specification
----------------------------------------
Scan_skew_group
---------------
Number of user-defined scan skew groups: 3
G1
G2
G3

1
```

To remove two of these scan skew groups, use the following command:

```
prompt> remove_scan_skew_group {G1 G3}
1
```

Now, only a single scan skew group remains:

```
prompt> report_scan_skew_group

****************************************
Report : Scan Skew Group
Design : top
Version: O-2018.06-SP5-VAL
Date   : Thu Aug 15 00:00:00 2019
****************************************

----------------------------------------
TEST MODE: all_dft
VIEW    : Specification
----------------------------------------
Scan_skew_group
---------------
Number of user-defined scan skew groups: 1
G2

1
```

SEE ALSO

set_scan_skew_group(2)
report_scan_skew_group(2)

remove_scenarios

Removes one or more scenarios in multi-mode-multi-corner analysis.

SYNTAX

status **remove_scenarios**
scenario_list
[-all]

Data Types

scenario_list collection

ARGUMENTS

scenario_list

Specifies the scenarios to remove.

-all

Removes all scenarios. This option is mutually exclusive with the *scenario_list* argument and you must specify **-all** or *scenario_list*.

DESCRIPTION

This command deletes the given scenarios, or all scenarios if the *-all* option is given. Note that this command does not delete the mode or corner for the scenario.

The command returns the number of scenarios that were removed.

SEE ALSO

create_scenario(2)
remove_corners(2)
remove_modes(2)
report_scenarios(2)

remove_sdc

Remove constraints from modes, corners, or scenarios

SYNTAX

string **remove_sdc**
[-design]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-include *include_list*]
[-exclude *exclude_list*]

Data Types

mode_list list
corner_list list
scenario_list list
include_list list
exclude_list list

ARGUMENTS

-design

Remove constraints not associated with any mode, corner, or scenario

-modes *mode_list*

Remove constraints on these modes

-corners *corner_list*

Remove constraints on these corners

-scenarios *scenario_list*

Remove constraints on these scenarios

-include *include_list*

Specifies the types of data to be removed.

Valid values are **case_analysis**, **clock**, **clock_gating_check**, **clock_group**, **clock_latency**, **clock_sense**, **clock_transition**, **clock_uncertainty**, **data_check**, **data_check_ignore**, **disable_timing**, **drive_info**, **edrc**, **exception**, **file_line_info**,

ideal_network, **input_delay**, **load**, **max_time_borrow**, **output_delay**, **path_group**, **pvt**, and **timing_derate**.

By default, all types of data are removed.

This option cannot be specified with the **-exclude** option.

-exclude ***exclude_list***

Specifies the types of data to exclude from removal.

Valid values are **case_analysis**, **clock**, **clock_gating_check**, **clock_group**, **clock_latency**, **clock_sense**, **clock_transition**, **clock_uncertainty**, **data_check**, **data_check_ignore**, **disable_timing**, **drive_info**, **edrc**, **exception**, **file_line_info**, **ideal_network**, **input_delay**, **load**, **max_time_borrow**, **output_delay**, **path_group**, **pvt**, and **timing_derate**.

By default, all types of data are removed.

This option cannot be specified with the **-include** option.

DESCRIPTION

Removes SDC constraints from designs, corners, scenarios, or the design as a whole. Constraints (including clocks) are removed from the specified modes, corners, or scenarios, but the actual modes, corners, or scenarios are **not** deleted.

If no options are given, all SDC constraints are removed from all modes, corners, and scenarios. If the **-design** option is given, the constraints associated with the design as a whole (i.e. not associated with any particular mode, corner, or scenario) are removed. If the **-modes** option is given, all constraints associated with the given mode(s) are removed. Any scenario-based constraints referring to the mode's clocks will also be deleted. If the **-corners** option is given, all constraints associated with the given corner(s) are removed. If the **-scenarios** option is given, all constraints associated with the given scenario(s) are removed.

The analysis settings are cleared for all affected scenarios.

Note that when no options are given, the behavior will be equivalent to:

```
remove_sdc -design -modes [all_modes] -corners [all_corners] -scenarios [all_scenarios]
```

Multicorner-Multimode Support

By default, this command works on all scenarios. To specify different scenarios, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

In the following example, the constraints of mode mode1 and scenarios mode1_c1 and mode1_c2 are removed.

```
prompt> remove_sdc -modes mode1 -scenarios {mode1_c1 mode1_c2}
1
```

SEE ALSO

remove_modes(2)
remove_corners(2)
remove_scenarios(2)
remove_clock(2)
remove_ideal_latency(2)
remove_ideal_network(2)
remove_ideal_transition(2)
report_constraint(2)
reset_design(2)
reset_path(2)

remove_secondary_pg_placement_constraints

Remove specific secondary pg placement constraints that will allow dual rail cells to be inserted into specific regions.

SYNTAX

status **remove_secondary_pg_placement_constraints**
[-all]
[*constraint_name_list*]

Data Types

constraint_name_list list of string

ARGUMENTS

-all

Remove all secondary pg constraints.

constraint_name_list

Remove the specific secondary pg constraints.

DESCRIPTION

command will remove the secondary power placement constraint previously specified with create_secondary_pg_placement_constraints. One of the following two specifications must be given. If -all is specified, all the secondary power placement constraints previously specified with create_secondary_pg_placement_constraints will be removed. Otherwise, the constraints specified by the list of constraint names will be removed. VA shapes are updated to reflect the state with the constraint(s) removed when the remove_secondary_pg_placement_constraints is followed by commit_secondary_pg_placement_constraints.

EXAMPLES

The following example removes the pg constraint named pg_cstr0.

prompt> **remove_secondary_pg_placement_constraints pg_cstr0**

The following example removes all pg constraints.

```
prompt> remove_secondary_pg_placement_constraints -all
```

SEE ALSO

commit_secondary_pg_placement_constraints(2)
create_secondary_pg_placement_constraints(2)
report_secondary_pg_placement_constraints(2)

remove_sense

Removes unateness information defined on pins or cell timing arcs.

SYNTAX

string **remove_sense**
[-type *type*]
[-all]
[-clocks *clock_list*]
object_list

Data Types

type string
clock_list list
object_list list

ARGUMENTS

-type *type*

Specifies whether the type of sense being removed refers to clock networks or data networks. The possible values for the type variable are: 'clock', 'data'. Note that 'clock' is assumed to be the default if **-type** option is not given.

-clocks *clock_list*

Optionally specifies a list of clock objects to be associated with the given pin objects in object_list. If the **-clocks** option is specified, only the unateness specified for that particular clock domain will be removed. Otherwise, unateness information for all clocks passing through the given pin objects will be removed. The **-clocks** option can only remove clock sense predefined by **set_sense -clock**. It does not remove the default clock sense setting for this given pin.

-all

Remove all unateness information in current design.

object_list

Lists of pins or cell timing arcs with predefined unateness to remove.

DESCRIPTION

Removes the unateness information predefined by the user. To set the unateness information, use the **set_sense** command.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example removes positive unateness defined on a pin named **XOR/Z** with respect to clock **CLK1**, but does not remove the negative unateness.

```
prompt> set_sense -positive -clocks [get_clocks CLK1] XOR/Z
prompt> set_sense -negative XOR/Z
prompt> remove_sense -clocks [get_clocks CLK1] XOR/Z
```

The following example removes all unateness information in the current design.

```
prompt> remove_sense -all
```

SEE ALSO

set_sense(2)

remove_shape_patterns

Removes shape_patterns from the current design.

SYNTAX

int **remove_shape_patterns**
[-force]
[-verbose]
shape_pattern_list

Data Types

shape_pattern_list collection

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Display verbose deletion information.

shape_pattern_list

Specifies a list of shape_patterns to remove. The list may contain shape_pattern collections, specified using the **get_shape_patterns** command.

DESCRIPTION

This command removes all specified shape_patterns. Shape_patterns are disassociated from their nets, layers, edit groups and groups if any.

The number of removed shape_patterns are returned.

EXAMPLES

The following example removes shape_patterns:

```
prompt> set a [get_shape_patterns -of_objects [get_nets VDD]]
{"PATH_PATTERN_18_06", "PATH_PATTERN_31_256", "PATH_PATTERN_28_122", "PATH_PATTERN_31_255"}

prompt> remove_shape_patterns $a
4
```

The following example removes shape_patterns specified by a pattern list:

```
prompt> remove_shape_patterns POLYGON_PATTERN_28_* -verbose
Removed shape_pattern POLYGON_PATTERN_28_173
Removed shape_pattern POLYGON_PATTERN_28_174
2
```

SEE ALSO

get_shape_patterns(2)

remove_shapes

Removes shapes from the current design.

SYNTAX

int **remove_shapes**
[-force]
[-verbose]
shape_list

Data Types

shape_list collection

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Display verbose deletion information.

shape_list

Specifies a list of shapes to remove. The list may contain shape collections, specified using the **get_shapes** command.

DESCRIPTION

This command removes all specified shapes. Shapes are disassociated with their nets, ports, terminals, layers, and edit groups if any.

When a terminal shape is removed, the associated terminal is removed as well. A terminal cannot exist without its shape.

When a port shape is removed, the associated terminal is removed as well. However the port is not removed, because a port may have zero or more terminals.

The number of removed shapes are returned.

EXAMPLES

The following example removes shapes:

```
prompt> set a [get_shapes -of_objects n300
{"PATH_18_1906", "PATH_31_256", "PATH_28_1322", "PATH_31_255"}

prompt> remove_shapes $a
4
```

The following example removes shapes specified by a pattern list:

```
prompt> remove_shapes POLYGON_28_* -verbose
Removed shape POLYGON_28_1738
Removed shape POLYGON_28_1740
2
```

SEE ALSO

get_shapes(2)

remove_shaping_blockages

Removes the specified shaping blockages.

SYNTAX

status **remove_shaping_blockages**
[-force]
[-verbose]
-all | *blockage_list*

Data Types

blockage_list list

ARGUMENTS

-force

Force the removal of the shaping blockage, even if the shaping blockage is locked. If this option is not provided and any of the affected objects has locked status, the command fails and generates a Tcl error.

-verbose

Show detailed information while removing shaping blockages.

-all

Removes all shaping blockages from the current design. This option is mutually exclusive with the *blockage_list* argument, but you must specify one.

blockage_list

Specifies the shaping blockages. You can specify a collection or the patterns by using the following formats:

* A collection of shaping blockages
* An asterisk (*), which indicates all shaping blockages
* A string

For example, specify a "*blockage_list*" of {SB_1234} to match the shaping blockage named SB_1234.

DESCRIPTION

This command removes the specified shaping blockages.

EXAMPLES

The following example removes all shaping blockages, even if some shaping blockages are locked.

```
prompt> remove_shaping_blockages -all -force
1
```

SEE ALSO

create_shaping_blockage(2)
get_shaping_blockages(2)

remove_shaping_channels

Removes a list or collection of channels from the current block.

SYNTAX

int **remove_shaping_channels**
channel_objects

Data Types

channel_objects list or collection

ARGUMENTS

channel_objects

Specifies the list or collection of channels to remove from the current block.

DESCRIPTION

This command removes a list or collection of channels. Any channel can be removed, even if it is assigned to a shaping_constraint object. Removing a channel might cause a shaping_constraint object to become unset. In that case, the channel removal also triggers a removal of the shaping_constraint object.

EXAMPLES

The following command removes a channel.

prompt> **remove_shaping_channels OBJECT_CHANNEL_0**

The following command removes all channels associated with the given shaping_constraint object.

prompt> **remove_shaping_channels DEFAULT_VA/SHAPING_CONSTRAINT_0_external_channels/***

The following command removes all channels associated with the given constrainable object.

prompt> **remove_shaping_channels DEFAULT_VA/***

SEE ALSO

create_shaping_channel(2)
get_shaping_channels(2)
channel(3)

remove_shaping_constraints

Removes a list of shaping_constraint objects.

SYNTAX

int **remove_shaping_constraints**
shaping_constraint_objects

Data Types

shaping_constraint_objects list or collection

ARGUMENTS

shaping_constraint_objects

A list or collection of shaping_constraint objects to be removed from the current block.

DESCRIPTION

Removes a list or collection of shaping_constraint objects from the current block. Note that shaping_constraint objects of type boundary_status cannot be removed by this (or any) command as they are derived constraints and as such the tool automatically creates and removes them as needed.

EXAMPLES

The following command removes a specific shaping_constraint.

prompt> **remove_shaping_constraints */SHAPING_CONSTRAINT_0_target_location**

The following command removes all shaping_constraints for the object u1/u2.

prompt> **remove_shaping_constraints u1/u2/***

SEE ALSO

create_shaping_constraint(2)
get_shaping_constraints(2)
shaping_constraint(3)

remove_shield_association

Disassociates shielding shapes and vias from shielded nets.

SYNTAX

status **remove_shield_association**
[-objects *shape_and_via_list*]
[-nets *net_list*]

Data Types

shape_and_via_list string, list or collection
net_list string, list or collection

ARGUMENTS

-objects *shape_and_via_list*

Specifies a list of shielding shapes and vias to remove from shield association.

-nets *net_list*

Specifies a list of shielded nets to remove from shield association.

DESCRIPTION

This command disassociates the specified shielding shapes and vias from the shielded nets. At least one of the **-objects** or **-nets** options must be specified. This command will return 1 even if this is no existing shield association between the specified shielding objects and shielded nets.

EXAMPLES

The following example removes shielding shape 'shape1' and shielded net 'net1' from shield association.

```
prompt> remove_shield_association -objects shape1 -nets net1
1
```

The following example removes shielding shape 'shape1' from all shield associations.

```
prompt> remove_shield_association -objects shape1
1
```

The following example removes shielding via 'via1' from all shield associations.

```
prompt> remove_shield_association -objects via1
1
```

The following example removes shielded net 'net1' from all shield associations.

```
prompt> remove_shield_association -nets net1
1
```

SEE ALSO

add_shield_association(2)

remove_signal_io_constraints

Removes signal I/O constraints from the specified I/O guides.

SYNTAX

status **remove_signal_io_constraints**
[-io_guide_list *io_guide*]

Data Types

io_guide list

ARGUMENTS

-io_guide_list

Specifies the name or collection of I/O guides. If no I/O guides are specified, signal constraints on all I/O guides are removed.

DESCRIPTION

This command removes signal constraints from I/O guides.

EXAMPLES

The following example removes signal constraints from an I/O guides io_guide1 and io_guide2.

prompt> **remove_signal_io_constraints -io_guide_list {io_guide1 io_guide2}**

The following example removes signal constraints from all I/O guides.

prompt> **remove_signal_io_constraints**

SEE ALSO

place_io(2)
report_signal_io_constraints(2)
set_signal_io_constraints(2)

remove_site_arrays

Removes site arrays from the current design.

SYNTAX

int **remove_site_arrays**
[-force]
[-verbose]
-all | *site_array_list*

Data Types

site_array_list collection

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Display verbose deletion information.

-all

Removes all site arrays in the current design.

site_array_list

Specifies a list of site arrays to remove. The list may contain site array collections, specified using the **get_site_arrays** command.

DESCRIPTION

This command removes all specified site arrays. Site arrays are disassociated from their environment like stack of site arrays.

Removal of a site array will result in change in stack order and may affect site rows of other site arrays if the removed site array is opaque.

The number of removed site arrays is returned.

EXAMPLES

The following example removes site arrays:

```
prompt> set a [get_site_arrays SA*]
{"SA1", "SA2_4", "SA3"}

prompt> remove_site_arrays $a
3
```

The following example removes site arrays specified by a pattern list:

```
prompt> remove_site_arrays SA* -verbose
Removed site array SA1
Removed site array SA2_4
Removed site array SA3
3
```

SEE ALSO

create_site_array(2)
get_site_arrays(2)

remove_site_defs

Removes site defs from the tech of current or specified library.

SYNTAX

int **remove_site_defs**
[-library *library*]
[-tech *tech_object*]
[-verbose]
-all | *site_def_list*

Data Types

library string or collection
tech_object collection
site_def_list string or collection

ARGUMENTS

-library *library*

Removes the site definition from the technology of this library. Default is current library's tech. This is mutually exclusive with the **-tech** option.

-tech *tech_object*

Removes the site definition from this technology. Default is current library's tech. This is mutually exclusive with the **-library** option.

-verbose

Display verbose deletion information.

-all

Removes all site defs in the tech of current or specified library.

site_def_list

Specifies a list of site defs to remove. The list can contain site def collections, specified using the **get_site_defs** command.

DESCRIPTION

This command removes all specified site defs. Site defs are disassociated from their site_row and site_array usages.

Removal of a site def will result in unbound site_row and site_array objects.

The number of removed site defs is returned.

EXAMPLES

The following example removes site defs:

```
prompt> set a [get_site_defs SD*]
{"SD1", "SD2_4", "SD3"}

prompt> remove_site_defs $a
3
```

The following example removes site defs specified by a pattern list:

```
prompt> remove_site_defs SD* -verbose
Removed site def SD1
Removed site def SD2_4
Removed site def SD3
3
```

SEE ALSO

create_site_def(2)
get_site_defs(2)
report_site_defs(2)

remove_site_rows

Removes site rows from the current design.

SYNTAX

int **remove_site_rows**
[-force]
[-verbose]
-all | *site_row_list*

Data Types

site_row_list collection

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Display verbose deletion information.

-all

Removes all site rows in the current design.

site_row_list

Specifies a list of site rows to remove. The list may contain site row collections, specified using the **get_site_rows** command.

DESCRIPTION

This command removes all specified site rows. Site rows are disassociated from their environment.

Removal of a site row will result in change in core area of block.

The number of removed site rows is returned.

EXAMPLES

The following example removes site rows:

```
prompt> set a [get_site_rows SR*]
{"SR1", "SR2_4", "SR3"}

prompt> remove_site_rows $a
3
```

The following example removes site rows specified by a pattern list:

```
prompt> remove_site_rows SR* -verbose
Removed site row SR1
Removed site row SR2_4
Removed site row SR3
3
```

SEE ALSO

create_site_row(2)
get_site_rows(2)

remove_skew_macros

Removes existing specifications defined by the **set_skew_macros** command.

SYNTAX

int **remove_skew_macros**
[-bank_name *name*]

Data Types

name string

ARGUMENTS

-bank_name *name*

Specifies the bank whose **set_skew_macros** specification needs to be removed. If no banks are specified, then all the **set_skew_macros** specifications on all banks are removed.

DESCRIPTION

The **remove_skew_macros** command removes existing specifications set by using the **set_skew_macros** command. If the bank_name is not specified, the command removes all **set_skew_macros** specifications.

EXAMPLES

The following example removes the bank named 'data1' specification set with the **set_skew_macros** command.

prompt> **remove_skew_macros -bank_name data1**

The following example removes **set_skew_macros** specifications on all banks.

prompt> **remove_skew_macros**

SEE ALSO

report_skew_macros(2)
set_skew_macros(2)

remove_stdcell_fillers_with_violation

Deletes filler cells that have routing DRC violations.

SYNTAX

status **remove_stdcell_fillers_with_violation**
[-name *cell_name*]
[-check_only true | false]
[-check_between_fixed_objects true | false]
[-boundary *rectangular_regions*]
[-post_eco true | false]
[-shorts_only true | false]

Data Types

cell_name list
rectangular_regions list

ARGUMENTS

-name *cell_name*

Specifies the cell instances to consider for removal. The names can include wildcard characters.

If you do not specify this option (or if you specify '*'), the name defaults to '*xofiller*'. If you use this default pattern, all instance names that have the 'xofiller' substring in them, including hierarchical names, are considered for removal.

If the specified cells are not filler cells (which have connections to signal nets), a warning message alerts you about the cells that are not deleted.

-check_only true | false

Controls whether the command runs in checking mode or removal mode.

If **false** (the default), the command runs in removal mode.

If **true**, the command runs in checking mode and does not actually remove the filler cells. In checking mode, the command checks for violations and outputs a file named <*current_design_name*>_fillers_with_violation.rpt that contains a list of the filler cells to be removed with the first violation for each filler cell. This output file is overwritten every time you run the command with the **-check_only true** option.

-check_between_fixed_objects true | false

Controls which objects are checked for violations.

If **false** (the default), the command checks only for violations between filler objects and top-level signal routing objects.

If **true**, the command checks for violations between filler objects and all neighboring objects, including signal routing objects, filler cells, standard cells, PG rail objects, terminals, and so on.

-boundary ***rectangular_regions***

Specifies the rectangular regions within which to delete filler cells with violations. Use the following syntax to specify each rectangular region:

{{*llx lly*} {*urx ury*}}

The *llx* and *lly* arguments specify the x- and y-coordinates of the lower-left corner of the rectangular region; the *urx* and *ury* arguments specify the x- and y-coordinates of the upper-right corner of the rectangular region. The units of the coordinates are the main library units. You can specify one or more rectangular regions.

If you do not specify this option, the command deletes filler cells with violations for the entire design.

-post_eco true | **false**

Controls whether the command deletes only filler cells marked as post-ECO cells by the **create_stdcell_fillers** command.

If **false** (the default), the command deletes filler cells with violations for the entire design.

If **true**, the command deletes only filler cells with violations marked as post-ECO cells by the **create_stdcell_fillers -post_eco** command. After filler cell removal, the command removes the post-ECO markings on the remaining filler cells. The runtime depends on the distribution of the added filler cells; if they are scattered across the design, the runtime can increase significantly.

This option is mutually exclusive with the **-name** and **-boundary** options.

-shorts_only true | **false**

Controls whether the command deletes filler cells with short violations only.

If **false** (the default), the command deletes filler cells with all violations.

If **true**, the command deletes filler cells with short violations only.

This option is mutually exclusive with the **-check_only** option.

DESCRIPTION

This command deletes the specified filler cells that have routing DRC violations. Filler cells are standard cells that do not have any connections to signal nets. The command does not remove filler cells whose **physical_status** attribute has a value of **fixed** or **locked**, even if they have violations. To remove these filler cells, you must first set their **physical_status** attribute to **placed**.

This command is useful when you add filler cells after performing signal routing.

The command deletes filler cells that have power or ground (PG) pins but are not assigned to a PG net if they touch other PG objects, such as rails or pins assigned to a PG net. Therefore, you should connect these pins to PG nets before running the **remove_stdcell_fillers_with_violation** command. For information about connecting pins to PG nets, see the man pages for the **connect_pg_net** and **create_stdcell_fillers** commands.

The command does not remove filler cells that have internal violations, which are violations between objects of the same cell; however, it does issue a warning if it detects this type of violation.

The command outputs an information message about the violation type that causes the deletion for the first 50 deleted cells. If there are multiple violations for the same cell, only one of them is listed.

It is a known limitation of the **remove_stdcell_fillers_with_violation** command that when filler cells are removed, new violations

might be introduced with neighboring filler cells. Therefore, you should run the **remove_stdcell_fillers_with_violation** command multiple times until no more filler cells are removed. In other words, until the command issues the "Deleted 0 cell instances" message.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all filler cells with the prefix "xofiller" that have routing DRC violations. Before removing the filler cells, their status is set to unfixed.

```
prompt> change_selection [get_cells -quiet {xofiller*}]
prompt> set_locked_objects [get_selection] -unlock
prompt> change_selection [get_cells -quiet {xofiller*}] -remove
prompt> remove_stdcell_fillers_with_violation -name "xofiller*"
```

The following example removes the filler cells with routing DRC violations within the specified rectangular region.

```
prompt> remove_stdcell_fillers_with_violation -boundary { { {16.12 32.38} {18.28 34.66} } }
```

The following example shows the post-ECO filler cell insertion flow.

```
prompt> create_stdcell_fillers -post_eco \
  -lib_cells {mylib/DCAP_4X mylib/DCAP_2X mylib/DCAP_1X}
prompt> remove_stdcell_fillers_with_violation -post_eco true
```

The following example removes the filler cells with short violations only.

```
prompt> remove_stdcell_fillers_with_violation -shorts_only true
```

The following example reports all filler cells with routing DRC violations, but does not remove the filler cells.

```
prompt> remove_stdcell_fillers_with_violation -check_only true
```

The following is the content of the generated my_design_fillers_with_violation.rpt file:

```
****************************************
Report : fillers_with_violation
Design : my_design
Version: J-2014.06
Date   : Mon Jun 16 01:10:44 2014
****************************************

instance_name     master_name  violation_type
  layer  violation_location              all_fixed_shapes
xofiller!DCAP!26  DCAP         Less than minimum edge length
  M1-M1  (504.565 635.990  504.585 636.155)  yes
xofiller!DCAP!13  DCAP         Diff net spacing
  M1-M1  (302.075 235.245  302.105 235.410)  no
xofiller!DCAP!59  DCAP         Less than minimum edge length
  M1-M1  (602.075 235.990  602.105 236.155)  yes
xofiller!DCAP!22  DCAP         Short
  M1-M1  (204.565 216.245  204.585 216.410)  no
```

```
xofiller!DCAP!10  DCAP        Diff net spacing
  M1-M1  (102.075 415.245  102.105 415.410)  no.in -0.25in
```

In the report, the all_fixed_shapes column indicates whether the first violation is caused only by fixed shapes.

SEE ALSO

connect_pg_net(2)
create_stdcell_fillers(2)

remove_stub_chains

Removes stub chains from the current design.

SYNTAX

int **remove_stub_chains**
[-verbose]
[-all | *stub_chains*]

Data Types

stub_chains collection

ARGUMENTS

-verbose

Display verbose deletion information.

-all

Removes all stub chains in the current design.

stub_chains

Specifies a list of stub chains to remove. The list may contain stub chain collections, specified using the **get_stub_chains** command.

DESCRIPTION

This command removes all specified stub chains.

The number of removed stub chains is returned.

EXAMPLES

The following example removes stub chains:

```
prompt> set a [get_stub_chains STUB*]
{"STUB1", "STUB2", "STUB3"}

prompt> remove_stub_chains $a
3
```

The following example removes stub chains specified by a pattern list:

```
prompt> remove_stub_chains STUB* -verbose
Removed stub chain STUB1
Removed stub chain STUB2
Removed stub chain STUB3
3
```

SEE ALSO

create_stub_chain(2)
get_stub_chains(2)

remove_supernet_exceptions

Removes supernet transparent settings from the current block.

SYNTAX

status **remove_supernet_exceptions**
-pins *pins_list* | -cells *cells_list* | -all

Data Types

pins_list list
cells_list list

ARGUMENTS

-pins

Specifies the pins in the block whose supernet transparent markings are to be removed. The pins in the list must belong to the same instance.

-cells

Specifies the cells in the block whose supernet transparent settings are to be removed. If the specified cell is a repeater then its *-disable_cell* settings are removed. If the specified cell is non-repeater then supernet transparent settings on the pins are removed.

-all

Removes all the markings of supernet transparent from the block.

DESCRIPTION

This command removes cells and pins as being transparent for supernets from the block.

EXAMPLES

The following example removes all the supernet transparent pins of the instance cntrl/U166

```
prompt> remove_supernet_exceptions -cell {cntrl/U166}
```

SEE ALSO

report_supernet_exceptions(2)
set_supernet_exceptions(2)

remove_supernets

Remove a supernet.

SYNTAX

int **remove_supernets**
[-design *design*]
supernet_collection

ARGUMENTS

-design *design*

Specifies the design in which to remove the supernet. If no design is specified, the supernet is removed from the current design.

supernet_collection

Required argument. The supernets to be removed.

DESCRIPTION

This command allows the user to remove (i.e. delete) supernets. The specified supernets are removed.

EXAMPLES

The following example removes a supernet named IF1.

```
prompt> remove_supernets [get_supernets IF1]
```

SEE ALSO

get_supernets(2)
create_supernet(2)

remove_taps

Removes existing tap points from the design.

SYNTAX

status **remove_taps**
[*taps*]

Data Types

taps list or collection

ARGUMENTS

taps

Specifies a list of taps to be removed. The taps can be specified as name patterns or collections of taps. When this argument is not specified, the **remove_taps** command removes all of the taps from the design.

DESCRIPTION

The **remove_taps** command removes tap points that are previously created using the **create_taps** command. To remove specific taps, use the **get_taps** command to select the tap cells.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example shows how to remove all the taps that are currently defined in the design:

```
prompt> remove_taps

INFO: All session taps are removed
```

The following example shows how to remove particular taps:

prompt> **remove_taps Tap_2**

INFO: Removing the specified tap(Tap_2) successfully.

prompt> **remove_taps [get_taps Tap_3]**

INFO: Removing the specified tap(Tap_3) successfully.

SEE ALSO

create_taps(2)
get_taps(2)
report_taps(2)

remove_target_library_subset

Removes target library subsets from the top block and hierarchical cells.

SYNTAX

status **remove_target_library_subset**
[-objects *objects*]
[-top]

Data Types

objects list or collection

ARGUMENTS

-objects *objects*

Specifies the hierarchical cells from which to remove the target library subsets. The cells must be in the current block.

If you do not specify this option or the **-top** option, the tool uses the **-top** option by default. Specifying both options is also supported.

-top

Removes the target library subset from the top block.

If you do not specify this option or the **-objects** option, the tool uses the **-top** option by default. Specifying both options is also supported.

DESCRIPTION

This command removes the target library subsets from the top block and hierarchical cells. This command only removes the subsets which were explicitly set on the objects with the set_target_library_subset command. This command does not remove subsets which are inherited from an ancestor hierarchical cell.

EXAMPLES

The following example removes the target library subset from the top hierarchy.

```
prompt> remove_target_library_subset -top
```

The following example removes the target library subset from a hierarchical cell.

```
prompt> remove_target_library_subset -objects [get_cells top/mid]
```

SEE ALSO

set_target_library_subset(2)
report_target_library_subset(2)

remove_tech

Removes the given technology object from the library.

SYNTAX

string **remove_tech**
tech_object

Data Types

tech_object list

ARGUMENTS

tech_object

Provides a list of tech objects to remove.

DESCRIPTION

This command removes the given tech object from the library that contains it.

EXAMPLES

The following command removes the tech object from the current library.

prompt> **remove_tech [get_techs -of_objects [current_lib]]**
1

SEE ALSO

create_tech(2)
get_techs(2)

remove_terminals

Removes terminals from the current design.

SYNTAX

int **remove_terminals**
[-force]
[-verbose]
terminal_list

Data Types

terminal_list collection

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail and generate a Tcl error.

-verbose

Display verbose deletion information.

terminal_list

Specifies a list of terminals to remove. The list can contain terminal collections specified using the **get_terminals** command.

DESCRIPTION

This command removes all specified terminals. The terminal is disassociated from its port and its shape or via. The shape or via is no longer owned by any terminal.

The number of removed terminals are returned.

EXAMPLES

The following example removes terminals of a port.

```
prompt> set a [get_terminals -of_objects [get_ports port1]]
{"TM_1", "TM_2", "TM_3", "TM_4"}
prompt> remove_terminals $a
4
```

SEE ALSO

create_terminal(2)
get_terminals(2)

remove_test_model

Permanently removes the test model associated with a design.

SYNTAX

status **remove_test_model**
[-design *design_name*]

ARGUMENTS

-design *design_name*

Specifies the design from which to remove the test model. If this option is not specified, the current design is used.

DESCRIPTION

This command permanently removes the test model associated with a design. If no design name is specified, the test model for the current design is removed.

EXAMPLES

The following command removes the test model for the design named *my_design*:

prompt> **remove_test_model -design my_design**

SEE ALSO

list_test_models(2)
read_test_model(2)

remove_test_protocol

Removes a test protocol from memory for the current design.

SYNTAX

status **remove_test_protocol**

ARGUMENTS

The **remove_test_protocol** command has no arguments.

DESCRIPTION

The **remove_test_protocol** command removes the test protocol from the current design.

EXAMPLES

The following example removes the test protocol from the current design named *my_design*:

```
prompt> remove_test_protocol
1
```

SEE ALSO

create_test_protocol(2)
write_test_protocol(2)

remove_tie_cells

Removes tie cells from the current design.

SYNTAX

status **remove_tie_cells**
[-objects *object_name_or_collection*]

Data Types

object_name_or_collection collection or list

ARGUMENTS

-objects *object_name_or_collection*

Specifies the objects for tie cell removal. Currently supported object types are cell and lib_cell. If this option is not used, all tie cells in current design will be removed. If the object type is cell and it is a tie cell, only the cell will be removed. If the object type is lib_cell, and the lib_cell is a tie cell, all the instances of the lib_cell will be removed.

DESCRIPTION

This command removes tie cells from the design.

SEE ALSO

add_tie_cells(2)
set_lib_cell_purpose(2)

remove_timing_paths_disabled_blocks

Restore the reg2reg timing paths across sub-bocks previously ignored by command **set_timing_paths_disabled_blocks**.

SYNTAX

string **remove_timing_paths_disabled_blocks**

ARGUMENTS

N/A

DESCRIPTION

Restore the reg2reg timing paths across sub-blocks previously ignored by command **set_timing_paths_disabled_blocks**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

prompt> **remove_timing_paths_disabled_blocks**

SEE ALSO

set_timing_paths_disabled_blocks(2)
report_timing(2)
report_qor(2)

remove_topological_constraints

Removes the topological constraints from the current design.

SYNTAX

integer **remove_topological_constraints**
topological_constraints | -all

topological_constraints collection

ARGUMENTS

topological_constraints

Specifies a list of topological pin feedthrough constraints to be removed from the design. Each topological pin feedthrough constraint name in the *topological_constraints* must exist in the design.

You must specify the *topological_constraints* or **-all**. The *topological_constraints* and **-all** options are mutually exclusive.

-all

Removes all the topological pin feedthrough constraints in the design. The *topological_constraints* and **-all** options are mutually exclusive.

DESCRIPTION

This command removes the topological pin feedthrough constraints from the design. To create topological pin feedthrough constraints, use the **create_topological_constraint** command.

EXAMPLES

The following example removes all the topological pin feedthrough constraints specified by the \FITPF_CONSTRAINT_1*\FP collection in the current design.

```
prompt> get_topological_constraints TPF_CONSTRAINT_1*
{TPF_CONSTRAINT_1 TPF_CONSTRAINT_10 TPF_CONSTRAINT_11 TPF_CONSTRAINT_12}

prompt> remove_topological_constraints TPF_CONSTRAINT_1*
```

4

The following example removes all the remaining topological pin feedthrough constraints in the current design.

```
prompt> remove_topological_constraints -all
```

SEE ALSO

create_topological_constraint(2)
get_topological_constraints(2)
report_topological_constraints(2)

remove_topology_edges

Removes topology edges.

SYNTAX

int **remove_topology_edges**
[-all]
[-verbose]
[*topology_edge_list*]

Data Types

topology_edge_list list

ARGUMENTS

-all

Removes all topology_edges from the current block.

-verbose

Displays verbose deletion information.

topology_edge_list

Specifies a list of topology edges to remove. The list may contain topology edge names, patterns, or collections. A collection may be specified by using the **get_topology_edges** command.

RETURN VALUE

The number of removed topology edges.

DESCRIPTION

This command removes topology edges from the current block. Topology nodes that are endpoints of the edges at the time of removal are not themselves removed from the block, however, all topology repeaters owned by the removed edges are also

removed.

EXAMPLES

The following example removes topology_edges that match the pattern "TOPOLOGY_EDGE*".

```
prompt> remove_topology_edges TOPOLOGY_EDGE*
3
```

The following example removes all topology edges in the block.

```
prompt> remove_topology_edges -all
5
```

SEE ALSO

create_topology_edge(2)
create_topology_repeater(2)
get_topology_edges(2)
get_topology_repeaters(2)
remove_topology_repeaters(2)
report_topology_plans(2)

remove_topology_nodes

Removes topology nodes.

SYNTAX

int **remove_topology_nodes**
[-all]
[-join_edges]
[-verbose]
[*topology_node_list*]

Data Types

topology_node_list list

ARGUMENTS

-all

Removes all topology nodes from the current block.

-join_edges

After removing the node, set the start nodes of the removed node outbound edges to the node driving its inbound edge.

-verbose

Displays verbose deletion information.

topology_node_list

Specifies a list of topology nodes to remove. The list may contain topology node names, patterns, or collections. A collection may be specified by using the **get_topology_nodes** command.

RETURN VALUE

The number of removed topology nodes.

DESCRIPTION

This command removes topology nodes from the current block. Any topology_edges that are connected to topology nodes at the time of removal are not themselves removed from the block, rather their endpoints are disconnected from the topology nodes and remain disconnected.

EXAMPLES

The following example removes topology nodes that match the pattern "TOPOLOGY_NODE*".

```
prompt> remove_topology_nodes TOPOLOGY_NODE*
3
```

The following example removes all topology nodes in the block.

```
prompt> remove_topology_nodes -all
5
```

SEE ALSO

create_topology_edge(2)
create_topology_node(2)
get_topology_edges(2)
get_topology_nodes(2)
remove_topology_edges(2)
report_topology_plans(2)

remove_topology_plans

Removes topology plans.

SYNTAX

int **remove_topology_plans**
[-all]
[-verbose]
[*topology_plan_list*]

Data Types

topology_plan_list list

ARGUMENTS

-all

Removes all topology_plans from the current block.

-verbose

Displays verbose deletion information.

topology_plan_list

Specifies a list of topology plans to remove. The list may contain topology plan names, patterns, or collections. A collection may be specified by using the **get_topology_plans** command.

RETURN VALUE

The number of removed topology plans.

DESCRIPTION

This command removes topology_plans from the current block. All topology_nodes and topology_edges owned by the removed plans are also removed.

EXAMPLES

The following example removes topology_plans that match the pattern "TOPOLOGY_PLAN*".

```
prompt> remove_topology_plans TOPOLOGY_PLAN*
3
```

The following example removes all topology_plans in the block.

```
prompt> remove_topology_plans -all
5
```

SEE ALSO

get_topology_plans(2)
create_topology_plan(2)
current_topology_plan(2)
report_topology_plans(2)

remove_topology_repeaters

Removes topology repeaters.

SYNTAX

int **remove_topology_repeaters**
[-all]
[-verbose]
[*topology_repeater_list*]

Data Types

topology_repeater_list list

ARGUMENTS

-all

Removes all topology repeaters from the current block.

-verbose

Displays verbose deletion information.

topology_repeater_list

Specifies a list of topology repeaters to remove. The list may contain topology repeater names, patterns, or collections. A collection may be specified by using the **get_topology_repeaters** command.

RETURN VALUE

The number of removed topology repeaters.

DESCRIPTION

This command removes topology repeaters from the current block. topology_nodes that are endpoints of the repeaters at the time of removal are not themselves removed from the block, however all topology repeaters owned by the removed repeaters are also

removed.

EXAMPLES

The following example removes topology repeaters that match the pattern "TOPOLOGY_repeater*".

```
prompt> remove_topology_repeaters TOPOLOGY_repeater*
3
```

The following example removes all topology_repeaters in the block.

```
prompt> remove_topology_repeaters -all
5
```

SEE ALSO

create_topology_repeater(2)
create_topology_repeater(2)
get_topology_repeaters(2)
get_topology_repeaters(2)
remove_topology_repeaters(2)
report_topology_plans(2)

remove_track_constraint

This command removes a set of existing track constraints.

SYNTAX

int **remove_track_constraint**
[-all]
[-block *block*]
[-layer *layer*]
[-mask *masks*]
[-track_direction horizontal | vertical]

Data Types

block string
layer string
masks list of masks

ARGUMENTS

-all

The -all argument allows for the removal of all defined track constraints in the given block. It may not be used with any of the -layer, -mask, or -track_direction arguments.

-block *block*

The -block argument specifies the block on which to remove the track constraints. If no block is specified then the current block is used.

-layer *layer*

The -layer argument specifies the layer for which the track constraints should be removed.

-mask *masks*

The -mask argument specifies the mask(s) for which corresponding constraints will be removed. Giving an individual mask value such as mask_one will only remove constraints associated with mask_one. Giving a list of mask values will remove constraints associated with any of the mask values in the list.

Omitting the -mask argument will remove constraints associated with un-colored tracks.

-track_direction horizontal | vertical

The -track_direction argument specifies the track orientation for which corresponding constraints will be removed. Specifying either "horizontal" or "vertical" will remove track constraints associated with the given direction. Omitting this argument will

remove track constraints associated with both track directions.

DESCRIPTION

The remove_track_constraint command removes existing track constraints in the given block.

The user may choose to remove track constraints by either giving the -all argument or a combination of the -layer, -mask, and -track_direction arguments. In the latter case, the three arguments specify the exact constraints to be removed. The arguments specify in an exclusive manner rather than an inclusive manner, meaning that in order to be removed, the constraints must match the layer AND the mask(s) AND the direction.

EXAMPLES

The following command removes the constraints that apply to both vertical and horizontal tracks with a mask color of 1 on layer M1 in block current_block.

```
prompt> remove_track_constraint -block [current_block] -layer M1 -mask mask_one
```

SEE ALSO

set_track_constraint(2)
report_track_constraints(2)

remove_tracks

Removes tracks from the current design.

SYNTAX

status **remove_tracks**
-all | *track_list* | -layer *layer* [-dir X | Y]
[-force]
[-verbose]

Data Types

track_list list
layer collection of one item

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-all

Removes all tracks in the current design.

The *track_list*, **-all**, and **-layer** arguments are mutually exclusive; you must specify one of these arguments.

track_list

Specifies a list of tracks to remove. The list may contain track names, patterns, or collections. A collection may be specified by using the **get_tracks** command.

The *track_list*, **-all**, and **-layer** arguments are mutually exclusive; you must specify one of these arguments.

-layer *layer*

Specifies the routing layer from which to remove the track. You can specify only one layer, either by layer name, or a collection containing one layer.

The *track_list*, **-all**, and **-layer** arguments are mutually exclusive; you must specify one of these arguments.

-dir X | Y

Specifies the direction of the routing tracks to be removed. The valid values are **X** and **Y**.

By default, the direction is the routing direction of the layer specified in the physical library.

The option must be used with the **-layer** option.

-verbose

Prints additional messages.

DESCRIPTION

This command removes the specified tracks from the current design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes all tracks that are inside {{2 2} {25 25}}.

```
prompt> remove_tracks [get_tracks -within {{2 2} {25 25}}]
1
```

The following example removes all tracks that are located on the metal2 layer.

```
prompt> remove_tracks [get_tracks -of_objects METAL2] -verbose
Removing track TRACK_4683
1
```

The following example removes the routing tracks from the routing layer named m3 on the floorplan.

```
prompt> remove_tracks -layer m3
Warning: Direction is not specified. Using the layer preferred direction.
(NDMUI-125)
1
prompt> remove_tracks -layer m3 -dir X
1
```

SEE ALSO

create_track(2)
get_tracks(2)
report_tracks(2)

remove_utilization_configurations

Removes an existing utilization configuration.

SYNTAX

remove_utilization_configurations
[-scope *config_scope*]
-all | *config*

Data Types

config_scope	string
config	collection

ARGUMENTS

-scope *config_scope*

The values it can take are 'lib', 'tech' and 'block'. Restricts the scope of search for the configurations. By default, the order of search is block, lib and then tech i.e. the configuration is first searched in the current block, then the current library and finally the tech associated with the current library.

-all

Removes all existing configurations.

config

Specifies the name of the configuration to be deleted. A collection of utilization-configs can also be provided as input to this option.

DESCRIPTION

This command can be used to remove an existing utilization configuration.

EXAMPLES

The following example removes a utilization configuration with name 'config1'.

prompt> **remove_utilization_configurations config1**

The following example removes utilization configuration with name 'config2' stored in the current-design library (-scope lib).

prompt> **remove_utilization_configurations config2 -scope lib**

The following example removes all the utilization configurations in the current block.

prompt> **remove_utilization_configurations -all -scope block**

SEE ALSO

create_utilization_configuration(2)
get_utilization_configurations(2)
report_utilization(2)

remove_verification_priority

Removes the **verification_priority** attribute from the specified objects.

SYNTAX

status **remove_verification_priority**
[-all]
object_list

Data Types

object_list list

ARGUMENTS

-all

Removes the **verification_priority** attribute from all of the designs and instances in the designs.

object_list

Specifies a list of objects from which the attribute is to be removed.

DESCRIPTION

This command removes the **verification_priority** attribute from the specified objects.

EXAMPLES

The following example removes the **verification_priority** attribute from all of the designs:

prompt> **remove_verification_priority -all**

SEE ALSO

get_attribute(2)
list_attributes(2)
set_verification_priority(2)

remove_via_defs

Removes via defs from block or technology file.

SYNTAX

int **remove_via_defs**
[-design *design*]
[-library *library*]
[-tech *tech_object*]
[-verbose]
via_def_list

Data Types

design	collection
library	string or collection
tech_object	collection
via_def_list	string or collection

ARGUMENTS

-design *design*

Via definition will be removed from this block. Default is current block. This is mutually exclusive with -library and -tech option.

-library *library*

Via definition will be removed from the technology of this library. Default is current block. This is mutually exclusive with -library and -tech option.

-tech *tech_object*

Via definition will be removed from this technology. Default is current block. This is mutually exclusive with -library and -tech option.

-verbose

Display verbose deletion information.

via_def_list

Specifies a list of via defs to remove. The list may contain via def collections, specified using the **get_via_defs** command.

DESCRIPTION

This command removes all specified via defs. Via defs are disassociated from their via usages.

Removal of a via def will result in unbound via objects.

The number of removed via defs is returned.

EXAMPLES

The following example removes via defs:

```
prompt> set a [get_via_defs VD*]
{"VD1", "VD2_4", "VD3"}

prompt> remove_via_defs $a
3
```

The following example removes via defs specified by a pattern list:

```
prompt> remove_via_defs VD* -verbose
Removed via def VD1
Removed via def VD2_4
Removed via def VD3
3
```

SEE ALSO

create_via_def(2)
get_via_defs(2)
report_via_defs(2)

remove_via_ladder_constraints

Removes via ladder constraints for pins.

SYNTAX

status **remove_via_ladder_constraints**
[-pins *pins*]
[-all]

Data Types

pins collection

ARGUMENTS

-pins *pins*

Specifies the pins from which to remove the via ladder constraints. The pins must have block context and belong to the current block.

The command issues a warning message for each specified pin that does not have via ladder constraints.

This option is mutually exclusive with the **-all** option; you must specify one of these options.

-all

Removes the via ladder constraints from all pins.

This option is mutually exclusive with the **-pins** option; you must specify one of these options.

DESCRIPTION

The **remove_via_ladder_constraints** command removes the via ladder constraints from the specified pins.

This command returns 1 if it succeeds, 0 otherwise.

EXAMPLES

The following example removes the via ladder constraints for all pins.

```
prompt> remove_via_ladder_constraints -all
1
```

The following example removes the via ladder constraints for the specified pin.

```
prompt> remove_via_ladder_constraints -pins [get_pins u1/i2]
1
```

SEE ALSO

set_via_ladder_constraints(2)
report_via_ladder_constraints(2)
set_via_ladder_candidate(2)
reset_via_ladder_candidates(2)
report_via_ladder_candidates(2)

remove_via_ladder_rules

Removes via ladders rules for the block.

SYNTAX

status **remove_via_ladder_rules**

DESCRIPTION

This command removes via ladder rules for the block set using set_via_ladder_rules.

This command returns 1 if succeeded, 0 otherwise.

EXAMPLES

```
prompt> remove_via_ladder_rules
1
```

SEE ALSO

set_via_ladder_rules(2)
report_via_ladder_rules(2)

remove_via_ladders

Removes via ladder from the current design.

SYNTAX

int **remove_via_ladders**
[-verbose]
[-force]
[-nets *nets*]
[-pins *pins*]
[*via_ladder_list*]

Data Types

nets collection
via_ladder_list collection

ARGUMENTS

-force

Ignores the locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating a Tcl error.

-verbose

Displays verbose deletion information.

-nets *nets*

Removes all the via ladders which are connected to the specified net or collection of nets.

-pins *pins*

Removes all the via ladders which are connected to the specified pin or collection of pins.

via_ladder_list

Specifies a list of via ladders to be removed. The list may contain via ladder collections specified using the **get_via_ladders** command.

DESCRIPTION

This command removes all specified via ladders. All the shapes associated with the specified via ladders will be removed from the design.

The command returns the number of via ladders that were removed.

EXAMPLES

The following example removes a via ladder:

```
prompt> set a [get_via_ladder -of_objects MemData[31]]
{VIA_LADDER_0 VIA_LADDER_1}

prompt> remove_via_ladders $a
2
```

The following example removes via ladders specified by a pattern list:

```
prompt> remove_via_ladders -verbose *_1
Removed via ladder VIA_LADDER_1
1
```

SEE ALSO

get_via_ladders(2)

remove_via_mappings

Removes the via-mapping options used during redundant via insertion, from current block.

SYNTAX

status **remove_via_mappings**
[-from {*via_pattern*}]
[-to {*via_pattern*}]
[-all]

Data Types

via_pattern string

ARGUMENTS

-from {*via_pattern*}

Specifies the via-pattern corresponding to the vias to be replaced during redundant via insertion.

Any via-mapping whose "to be replaced" via matches with this pattern will be removed.

A via-pattern has two parts: viaDefName and site_spec mxn. viaDefName is via definition name. mxn specifies the number of contacts in the horizontal (m) and vertical (n) directions.

-to {*via_pattern*}

Specifies the via-pattern corresponding to the vias to be inserted during redundant via insertion.

Any via-mapping option whose "replacement" via matches with this pattern will be removed.

-all

Removes all the via-mapping options in the current-block.

DESCRIPTION

The **remove_via_mappings** command removes from the current block, any existing via-mapping options. One of the optional arguments must be specified.

EXAMPLES

The following example removes the via-mapping options for the replacement of vias with via-def 'VIA12':

prompt> **remove_via_mappings -from {VIA12 1x1}**

The following example removes the via-mapping options where, the replacement via is a double via with via-def 'VIA12':

prompt> **remove_via_mappings -to {VIA12 1x2}**

The following example removes the via-mapping options for via replacement from vias with via-def 'VIA12' to a double via with via-def 'VIA12':

prompt> **remove_via_mappings -from {VIA12 1x1} -to {VIA12 1x2}**

The following example removes all the existing via-mapping options in the current block:

prompt> **remove_via_mappings -all**

SEE ALSO

add_via_mapping(2)
report_via_mapping(2)

remove_via_matrixes

Removes via matrixes from the current design.

SYNTAX

int **remove_via_matrixes**
[-verbose]
[-force]
[via_matrix_list]

Data Types

via_matrix_list collection

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Display verbose deletion information.

via_matrix_list

Specifies a list of via matrixes to remove. The list may contain via matrix collections, specified using the **get_via_matrixes** command.

DESCRIPTION

This command removes all specified via matrixes. Via matrixes are disassociated with their nets, and edit groups if any.

The number of removed via matrixes are returned.

EXAMPLES

The following example removes via matrixes:

```
prompt> set a [get_via_matrixes -of_objects MemData[31]]
{VIA_MATRIX_1 VIA_MATRIX_0}

prompt> remove_via_matrixes $a
2
```

The following example removes via matrixes specified by a pattern list:

```
prompt> remove_via_matrixes -verbose *_1
Removed via matrix VIA_MATRIX_1
1
```

SEE ALSO

get_via_matrixes(2)

remove_via_regions

Removes user and system via regions from a frame block.

SYNTAX

int **remove_via_regions**
[-design *design*]
[-verbose]
[*via_regions*]

Data Types

design collection
via_regions collection

ARGUMENTS

-design *design*

Specifies the design for frame block from which to remove via regions. By default, the command removes via regions from the current frame block.

-verbose

Show detailed information while removing via regions.

via_regions

Specifies via regions to remove. The list may contain via region collections, specified using the **get_via_regions** command.

By default, the command removes all via regions from the frame block.

DESCRIPTION

This command removes all specified user and system via regions.

The number of removed via regions are returned.

EXAMPLES

The following example removes via regions:

```
prompt> set a [get_via_regions -of_objects Clk
{"Clk/VR_0", "Clk/VR_3"}

prompt> remove_via_regions $a
2
```

The following example removes vias specified by a pattern list:

```
prompt> remove_via_regions D1/*
2
```

SEE ALSO

create_via_region(2)
get_via_regions(2)
report_via_regions(2)

remove_via_rules

Removes via_rule objects from the block or technology file.

SYNTAX

collection **remove_via_rules**
[-design *design*]
[-library *library*]
[-tech *tech*]
[-all]
[*via_rule_list*]

Data Types

design	collection
library	collection
tech	collection
via_rule_list	string or collection

ARGUMENTS

-design *design*

Removes via rule the specified block. This option is mutually exclusive with the **-library** and **-tech** options. By default, via rules are removed from the current block.

-library *library*

Removes via rules from the technology for this library. This option is mutually exclusive with the **-tech** and **-design** options. By default, via rules are removed from the current block.

-tech *tech*

Removes via rules from the specified technology. This option is mutually exclusive with the **-library** and **-design** options. By default, via rules are removed from the current block.

-all

Removes all via rules in the specified scope. You must specify either **-all** or *via_rule_list*.

via_rule_list

Specifies a list of via rules to remove in the specified scope. You must specify either **-all** or *via_rule_list*.

RETURN VALUE

This command returns the removed via rule's count, an empty string if it fails, or a TCL_ERROR if there is a command syntax error.

DESCRIPTION

This command removes all specified via rules from the specified block or the technology file. Via rules are disassociated from their usages. You must specify either **-all** or *via_rule_list*.

EXAMPLES

The following example removes the via rule named *VR1*

```
prompt> remove_via_rules VR1
1
```

SEE ALSO

create_via_rule(2)
get_via_rules(2)
report_via_rules(2)

remove_vias

Removes vias from the current design.

SYNTAX

int **remove_vias**
[-force]
[-verbose]
via_list

Data Types

via_list collection

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Display verbose deletion information.

via_list

Specifies a list of vias to remove. The list may contain via collections, specified using the **get_vias** command.

DESCRIPTION

This command removes all specified vias. Vias are disassociated with their nets, ports, terminals, and edit groups if any.

When a terminal via is removed, the associated terminal is removed as well. A terminal cannot exist without its via.

When a port via is removed, the associated terminal is removed as well. However the port is not removed, because a port may have zero or more terminals.

The number of removed vias are returned.

EXAMPLES

The following example removes vias:

```
prompt> set a [get_vias -of_objects n300
{"VIA_S_1", "VIA_S_2", "VIA_SA_31", "VIA_C_43"}

prompt> remove_vias $a
3
```

The following example removes vias specified by a pattern list:

```
prompt> remove_vias VIA_S_* -verbose
Removed via VIA_S_1
Removed via VIA_S_2
2
```

SEE ALSO

get_vias(2)

remove_virtual_connections

Remove all virtual connections or specified virtual connections.

SYNTAX

status **remove_virtual_connections**
patterns
| *-all*

Data Types

ARGUMENTS

patterns

Remove matched virtual connection against patterns. Patterns can include the wildcard characters "*" and "?". Patterns can also include collections of type virtual connection. The two options are mutually exclusive with each other. You must specify one, but not both.

-all

Removes all the virtual connections. The two options are mutually exclusive with each other.

DESCRIPTION

This command removes all virtual connections or specified virtual connections.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples removes the virtual connection.

Remove all virtual connections.
prompt> **remove_virtual_connections -all**

Remove one virtual connection.

```
prompt> remove_virtual_connections \
    [get_virtual_connections -name SNPS_vc_2]

prompt> remove_virtual_connections SNPS_vc_2
```

Remove some virtual connections.

```
prompt> remove_virtual_connections \
    {SNPS_vc_2 SNPS_vc_3}

prompt> remove_virtual_connections SNPS_vc*

prompt> remove_virtual_connections \
    [get_virtual_connections -of_objects [get_pins {U212/ZN U256/I}]
```

SEE ALSO

place_eco_cells(2)
create_virtual_connection(2)
get_virtual_connections(2)
add_pins_to_virtual_connection(2)
remove_pins_from_virtual_connection(2)
get_attribute(2)
set_attribute(2)

remove_virtual_pads

Removes existing virtual pads.

SYNTAX

status **remove_virtual_pads**
[-net *net*]
[-coordinate {*x y*}]
[-layer *layer_name*]
[-all]

Data Types

net collection
x float
y float
layer_name collection

ARGUMENTS

-net *net*

Specifies the net from which to remove the virtual pads.

-coordinate *coordinate*

Specifies the coordinate of the virtual pads to remove.

-layer *layer_name*

Specifies the layer name on which to remove the virtual pads.

-all

Removes all virtual pads from the current design.

DESCRIPTION

This command removes existing virtual pads from the current design.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example removes the virtual pad for net VDD at location {1000 1000}.

prompt> **remove_virtual_pads -net VDD -coordinate {1000 1000}**

The following example removes all existing virtual pads.

prompt> **remove_virtual_pads -all**

SEE ALSO

analyze_power_plan(2)
read_virtual_pad_file(2)
report_virtual_pads(2)
set_virtual_pad(2)
write_virtual_pad_file(2)

remove_voltage_area_rules

Removes voltage area rules from the design.

SYNTAX

int **remove_voltage_area_rules**
[-verbose]
-all | *voltage_area_rules*

Data Types

voltage_area_rules collection

ARGUMENTS

-verbose

Displays verbose deletion information. Prints the name of each rule as it is deleted.

-all

Removes all voltage area rules in the design.

voltage_area_rules

Specifies the voltage area rules to remove. This can be a collection handle of voltage area rules or names of patterns. You can use the **get_voltage_area_rules** command to specify objects.

RETURN VALUE

The number of removed voltage area rules.

DESCRIPTION

This command removes voltage area rules from the design.

EXAMPLES

The following example removes the voltage area rule named "RULE1" from the design.

prompt> **remove_voltage_area_rules RULE1**

The following example removes all voltage area rules from the design.

prompt> **remove_voltage_area_rules -all**

SEE ALSO

create_voltage_area_rule(2)
get_voltage_area_rules(2)
report_voltage_area_rules(2)

remove_voltage_area_shapes

Removes voltage_area shapes from the current design.

SYNTAX

status **remove_voltage_area_shapes**
[-force]
[-verbose]
-all | *voltage_area_shape_list*

Data Types

voltage_area_shape_list list

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Prints additional messages.

-all

Removes all voltage_area shapes in the current design.

voltage_area_shape_list

Specifies the list of voltage_area shapes to be removed. The voltage_area_shape_list can be a collection handle of voltage_area_shapes or names of patterns. You can use the **get_voltage_area_shapes** command to specify objects.

RETURN VALUE

The number of removed voltage_area_shapes.

DESCRIPTION

This command removes voltage_area shapes from the design. If all voltage_area shapes of a particular voltage_area are removed, the voltage_area is removed as well.

EXAMPLES

The following example removes voltage_area shapes of voltage_area tap1:

```
prompt> remove_voltage_area_shapes [get_voltage_area_shapes -of_objects tap1]
3
```

The following example removes all voltage_area shapes in the design:

```
prompt> remove_voltage_area_shapes -all
10
```

SEE ALSO

create_voltage_area_shape(2)
get_voltage_area_shapes(2)
create_voltage_area(2)
get_voltage_areas(2)
report_voltage_areas(2)
remove_voltage_areas(2)

remove_voltage_areas

Removes voltage_areas from the current design.

SYNTAX

status **remove_voltage_areas**
[-force]
[-verbose]
-all | *voltage_area_list*

Data Types

voltage_area_list list

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-verbose

Prints additional messages.

-all

Removes all voltage_areas in the current design.

voltage_area_list

Specifies the voltage_areas. The voltage_area_list can be a collection handle of voltage_areas or names of patterns. You can use the **get_voltage_areas** command to specify objects.

RETURN VALUE

The number of removed voltage_areas.

DESCRIPTION

This command removes voltage_areas from the design.

EXAMPLES

The following example removes voltage_areas whose names begin with *tap*:

```
prompt> remove_voltage_areas tap*
1
```

SEE ALSO

create_voltage_area(2)
get_voltage_areas(2)
report_voltage_areas(2)
create_voltage_area_shape(2)
remove_voltage_area_shapes(2)
get_voltage_area_shapes(2)

rename

Renames or deletes a command.

SYNTAX

string **rename**
old_name
new_name

ARGUMENTS

old_name

Specifies the current name of the command.

new_name

Specifies the new name of the command.

DESCRIPTION

Renames the *old_name* command so that it is now called *new_name*. If *new_name* is an empty string, then *old_name* is deleted. The *old_name* and *new_name* arguments may include namespace qualifiers (names of containing namespaces). If a command is renamed into a different namespace, future invocations of it will execute in the new namespace. The **rename** command returns an empty string as result.

Note that the **rename** command cannot be used on permanent procedures. Depending on the application, it can be used on all basic builtin commands. In some cases, the application will allow all commands to be renamed.

WARNING: **rename** can have serious consequences if not used correctly. When using **rename** on anything other than a user-defined Tcl procedure, you will be warned. The **rename** command is intended as a means to wrap other commands: that is, the command is replaced by a Tcl procedure which calls the original. Parts of the application are written as Tcl procedures, and these procedures can use any command. Commands like **puts**, **echo**, **open**, **close**, **source** and many others are often used within the application. Use **rename** with extreme care and at your own risk. Consider using **alias**, Tcl procedures, or a private namespace before using **rename**.

EXAMPLES

This example renames my_proc to my_proc2:

```
prompt> proc my_proc {} {echo "Hello"}
prompt> rename my_proc my_proc2
prompt> my_proc2
Hello
prompt> my_proc
Error: unknown command 'my_proc' (CMD-005).
```

SEE ALSO

define_proc_attributes(2)

rename_block

Renames a block in memory to a new block in the same or different library, view or label.

SYNTAX

block_return **rename_block**
[-force]
[-verbose]
[-hierarchical]
[-from_block *block*]
-to_block *block_name*

Data Types

block_return Collection containing block or "".
block Block name in [libName:]blockName[/labelName][.view-Name] format or a collection of one block.
block_name Name in form [libName:]designName[/labelName][.viewName]

ARGUMENTS

-force

This option is used when the destination is (or may be) open and modified but not yet saved.

-verbose

Prints out additional debugging and informational messages.

-hierarchical

This option is used to rename an entire physical hierarchy of the given block. When used with a specific label name, it rename each non lib-cell sub-block and all its views into the specified label. It follows the hierarchy into the reference blocks and rename the sub-blocks referenced by its instances, stopping when it reaches the leaf-level lib-cell references. Each labeled block is renamed in-place in the library of the original block, and each of its non lib-cell instances would have their reference updated to point to the new labeled block.

-from_block *block*

Source block name with optional library , label, and view specifications. If the library specification is not present, the **current_lib** is used. If the label specification is not present, then the default un-named label is used. If the view specification is not present, then *design* is used. If the option is not given, the **current_block** is used.

-to_block *block_name*

The destination block name with optional library and view specifications. If the library, block, or view specifications are not

present, the values from the source block are used.

RETURN VALUE

Returns the block if successful, or O if not. If an illegal name is given for the source or destination block names, a Tcl error is raised.

DESCRIPTION

This command renames a source block to a destination with the same or different library or block in memory. If the source block is freshly created and has no on-disk copy, the in-memory block will just be renamed to the destination and the original block is gone. if the source block has a disk copy, it will be renamed in-memory only, while the on-disk copy is untouched. The renamed in-memory block will then be saved temporarily to disk so that it becomes independent from the source block (in case the source block is later removed). The temporarily saved disk copy will remain valid until a close_block **-force** command is issued to discard the change, or a **close_lib** command is used to close the library without saving the change, at which point the temporary design file and directory will be removed.

Note that the block cannot be renamed if any of its views are referenced by another blocks. For instance, in a top-mid-bot 3-level hierarchy setup, "top" itself can be renamed with the **rename_block** command, and the entire top-mid-bot hierarchy can be renamed with the **rename_block -hierarchical** command. But it is an error to individually rename "mid" or "bot" while "top" references them.

The command will fail if the destination library or block is opened in read-only mode, or if the destination block is opened and has been modified but not saved and the **-force** option is not used, or if an illegal name is given for the source or destination library, block or view.

When no view is explicitly specified for both source and destination then all available views of source block are renamed into destination block and the design view of destination block is returned.

A given source block view can't be renamed into a different destination block view, like a design view can't be renamed to an abstract view. An error is issued while trying to do such operation.

EXAMPLES

This example renames a current_block "Top" to "Top2". The destination library and view are taken from the original block.

```
prompt> current_block Top
prompt> rename_block -to_block Top2
{"Lib:Top2.design"}
prompt> current_block
{"Lib:Top2.design"}
```

This example renames a block Top into a different label Top/label1

```
prompt> rename_block -from_block Top -to_block Top/label1
{"Lib:Top/label1.design"}
```

This example renames the entire design hierarchy into a different label:

```
prompt> rename_block -from_block Top -to_block Top/label1 -hierarchical -verbose
Information: Renaming 'Lib:Top.design' to 'Lib:Top/label1.design'. (DES-055)
Information: Renaming 'Lib:Mid.design' to 'Lib:Mid/label1.design'. (DES-055)
Information: Renaming 'Lib:Mid.abstract' to 'Lib:Mid/label1.abstract'. (DES-055)
Information: Renaming 'Lib:Bot.design' to 'Lib:Bot/label1.design'. (DES-055)
Information: Renaming 'Lib:Bot.abstract' to 'Lib:Bot/label1.abstract'. (DES-055)
Information: Renaming 'Lib:Top.outline' to 'Lib:Bot/label1.outline'. (DES-055)
```

This example renames a block Mid into a new library RefLib1.

```
prompt> rename_block -from_block Mid -to_block RefLib1:
{"RefLib1:Mid.design"}
```

SEE ALSO

close_blocks(2)
copy_block(2)
create_block(2)
get_blocks(2)
get_designs(2)
open_block(2)
open_lib(2)
remove_blocks(2)
reopen_block(2)
save_block(2)

rename_module

Rename module(s) in memory.

SYNTAX

status **rename_module**
module_list
[*target_name*]
[-prefix *prefix_name*]
[-postfix *postfix_name*]

Data Types

module_list	list
target_name	string
prefix_name	string
postfix_name	string

ARGUMENTS

module_list

Specifies a list of modules to be renamed.

target_name

Specifies the new name of the module. *target_name* can only be used when *module_list* contains one module,

You must specify either *target_name*, or *-prefix*, or *-postfix*. These arguments are mutually exclusive; you can only specify one.

-prefix *prefix_name*

Specifies the prefix name that is inserted in front of names of modules in *module_list*. One of *target_name*, *-prefix*, or *-postfix* is required, but they are mutually exclusive.

-postfix *postfix_name*

Specifies the postfix name that is appended to the names of modules in *module_list*. One of *target_name*, *-prefix*, or *-postfix* is required, but they are mutually exclusive.

DESCRIPTION

The most simple use of this command is to assign a new name to a single module. Another use of this command is to add a prefix or postfix to multiple modules by using the -prefix or -postfix options.

Note that within a block, every module name must be unique and an error occurs if the name of the new module already exists.

EXAMPLES

The following example renames the module named *middle* to *mid*:

```
prompt> rename_module middle mid
Information: In design 'top', renaming module 'middle' to 'mid'.
1
```

The *-prefix* and *-postfix* options provide a convenient and efficient way to rename multiple modules. For instance, the following script prepends the string *NEW_* to the name of module *D*. The new module name will be reflected when querying for the reference name of the cells

```
prompt> get_cells -hierarchical -filter "ref_name == D"
{b_in_a/c_in_b/d1_in_c b_in_a/c_in_b/d2_in_c}

prompt> rename_module D -prefix NEW_
Information: In design 'myBlock', renaming module 'D' to 'NEW_D'. (NDMUI-623)

prompt> get_cells -hierarchical -filter "ref_name == D"   ;# no such cells!
prompt> get_cells -hierarchical -filter "ref_name == NEW_D"
{b_in_a/c_in_b/d1_in_c b_in_a/c_in_b/d2_in_c}
```

SEE ALSO

edit_module(2)
create_module(2)
remove_modules(3)

reopen_block

Changes the open mode of an open block and sets the top module of the block as current. The default open mode is "edit".

SYNTAX

return_val **reopen_block**
[-edit | -read | -new]
[-force]
[*block*]

Data Types

return_val 0 (or 1) if command failed (or succeeded).
block Block name in [libName:]blockName[/labelName][.viewName] format or a collection of one block.

ARGUMENTS

-edit

This is the default option when no open-mode is specified. It can be used to upgrade a block from read mode to edit mode. It can also be used to downgrade a block from new mode to edit mode, discard the in-memory block content and revert to the last saved state. It is an error to downgrade from new mode to edit mode if the block does not have a copy on disk.

-read

Downgrade a block from edit/new mode to read mode. Discard the in-memory block content and revert to the last saved state. Downgrade a block from edit/new mode to read mode. Discard the in-memory block content and revert to the last saved state. The block must have a copy on disk, otherwise an error message is printed. If the block has in-memory changes that haven't been saved to disk, the **-force** option must be used or an error message is printed.

-new

Discard the in-memory block content and reinitialize the block in new mode. If the block has in-memory changes that haven't been saved to disk, the **-force** option must be used or an error message is printed.

-force

When used with the **-read** option, force the block to be downgraded and reverted to the last saved state, even if the block has in-memory modifications that are not saved. When used with the **-new** option, force the block content to be discarded and become an empty block even if the block is modified but not yet saved.

block

The source block name with optional library, label, and view specifications. If the library specification is not present, the **current_lib** is used. If the label specification is not present, the default unnamed label is used. If the view specification is not

present, then *design* is used. If the option is not given, the **current_block** is used.

RETURN VALUE

Returns 1 if the block is saved and 0 if not. If an illegal name (with a slash) is given for the block, a Tcl error is raised.

DESCRIPTION

This command modifies the open mode of an already-open block. It is most often used to "downgrade" the open mode from edit to read, and can be used to reinitialize a block by re-opening it with the **-new** option. The command fails if you attempt to "downgrade" from edit or new mode to read mode, or from new mode to edit mode, and there is no copy on disk. The command also fails if you attempt to "downgrade" or reinitialize a block and it has been modified but not saved and the **-force** option is not used.

If the library is already open in read mode, **reopen_block -edit** will promote the library open mode to edit. This can be disabled by setting the **design.edit_read_only_libs** application option to false.

EXAMPLES

Reopen a read only Top to edit mode so it can be edited:

```
prompt> reopen_block -edit Top
1
```

Reopen a read only block, reinitialize it to re-create it:

```
prompt> reopen_block -new Top
1
```

Reopen an already-modified edit mode block into read mode, abandoning the changes along the way:

```
prompt> reopen_block -force -read Top
1
```

SEE ALSO

copy_block(2)
create_block(2)
current_block(2)
current_design(2)
get_blocks(2)
get_designs(2)
move_block(2)
open_block(2)
open_lib(2)

remove_blocks(2)
save_block(2)

reparent_cells

Moves one or more cells from their current hierarchy location to a new one.

SYNTAX

<new_cells> **reparent_cells**
[-remove_floating_pins]
[-port_prefix *prefix_name*]
[-cell_prefix *prefix_name*]
[-check_only]
-cells [*cell_list*]
-to *cell_path*

Data Types

cell_list list
cell_path string
prefix_name string
new_cells collection of moved cells

ARGUMENTS

-to *cell_path*

Specifies a destination to where the cells will be moved. As each object is moved, it carries with it its connectivity, which may result in port-punching and possibly port removal in the intervening levels between the object source location and the destination.

-cells *cell_list*

Specifies a list of hierarchical cells that will be moved from their current location to the destination given in the *-to* argument. Note that not all of the given cells must currently be in the same hierarchical path in the design.

-remove_floating_pins

If this option is given, when ports become unused as a result of the movement of the specified objects, they will be removed. By default, they are left alone so that the port signature of cells will match the original definition, possibly including additional ports due to the port-punching necessary when moving the cells.

-port_prefix *prefix_name*

When new ports are created during port-punching as cells are moved, they are given a default name which will match the net name of the original net (if possible, or that name with a suffix of _# to make the name unique). If the user would prefer a specific port prefix for the newly-created ports, they can specify it with this option.

-cell_prefix *prefix_name*

When the cells are moved, they are given a default name which will match the cell name of the original cell (if possible, or that name with a suffix of _# to make the name unique). If the user would prefer a specific cell prefix for the newly-created cells, they can specify it with this option. This can make it easier to identify moved cells at a later time.

-check_only

When this option is used, all of the pre-command checks are executed, and a report is given indicating which (if any) moves will be made. But nothing is moved, no ports are punched, no cells are uniquified. This option is used for script debugging to make sure the command will behave as desired.

DESCRIPTION

This command moves one or more cell from their current location in the logic hierarchy to a different place in the logic hierarchy. It preserves the connectivity of the cells during the move, which can result in new ports being created to carry the connectivity and keep it intact (this is called port-punching). Note that the specified cells need not be rooted at the same place in the design hierarchy (although they will most frequently be found together).

On successful completion, the command will destroy any input collection (as the moved cells no longer exist in their previous location(s)). The command returns a collection of the newly-moved cells, in the same order as they appeared in the one or more -cells options.

The command is undoable, and if undo is executed after the command completes, the entire system will be restored to the state it was at before the command executed. Note that many subsequent operations can clear the system undo stack, making it impossible to undo any previous commands, including this one. So exercise care when using this command.

If possible, the cell name will be kept the same after the move. Note that this is not the cell path, but the tail component of that path. If there is an already-existing cell with that name in the destination, a _# will be appended to the name in order to make it unique. If the -cell_prefix option is used it will be prepended to the cell name.

Note that before the command will execute, it will check that:

- All levels between and including the source(s) and destination are writable and editable
- Ports will be able to be made on all intermediate cells subject to dont_touch
- It is possible to update UPF as needed
- It is possible to update constraints as needed

If one or more of these checks fails the entire command will fail and report which of the above condition(s) are present and preventing the command from operating as specified.

This command currently operates on a gate-level netlist. When the gen_sys flow is in effect, the command operates on the post-elaboration (generic) netlist created in the system. In this case it will fail if one or more of the moved cells is not at or above the synthesis-invariant level (meaning that this move has no direct correspondent at the RTL level).

Other than the restrictions listed above, there are no restrictions in moving cells.

In order to keep the original port signature for each logic cell as intact as possible, by default any ports which become completely disconnected in reaction to the move will be left in place. If the -remove_floating_ports option is used, then those disconnected ports will be removed to leave the port signature as small as possible.

As is the case with all design-modification commands, in the case of multiply-instantiated modules, reparenting that affects one of a set of MIMs will cause a uniquification of the affected MIM(s) before the move takes place. Specifically, if an *instance* of a module that is multiply-instantiated is moved, no uniquification will (or needs to) take place. However, if an instance *inside* a multiply-instantiated module is moved, then the uniquification will take place.

As port-punching takes place, the command will attempt to preserve the names of the moved cells' internal nets as it creates new ports through the logic hierarchy. If there is a conflict at any given level with an already-existing port, the command will append _# to the net name, cycling through numbered additions until a port name with no conflict is found. If a user would prefer that the created ports have a specific prefix instead (perhaps to be able to identify punched ports later on), then the specified prefix will be prepended to the net name.

Also note that as the cells, both timing constraints and UPF are adjusted to account for the new structure. If the design is set to use the Golden UPF flow, the changes will be made consistent with that flow. SVF data will be updated to help Formality properly compute the correspondence of the modified design to previous versions. This helps underscore the economy of scale present when moving a cloud of connected logic all at the same time as opposed to moving the same set of cells one at a time.

EXAMPLES

The following example takes the classic top/mid/bot logic hierarchy, and moves mid1/bot2 into mid2.

```
prompt> reparent_cell mid1/bot2 -to mid2
{mid2/bot2_1}
```

Note that since there already was a mid2/bot2, that mid1/bot2 had _1 appended to it to make it unique and avoid the conflict with the other bot2.

The following example finds every cell called "glue_logic" and moves them to the top level.

```
prompt> reparent_cell [get_cells -hierarchical glue_logic] -to /
{glue_logic glue_logic_1 glue_logic_2 glue_logic_3}
```

Note that in this case since the command itself would have created cell name conflicts that the second and subsequent cells have _# appended to their name.

SEE ALSO

group_cells(2)
ungroup_cells(2)

replace_clock_gates

Automatically searches for and replaces all combinational cells that are clock gates with integrated latch-based clock gate cells.

SYNTAX

status **replace_clock_gates**

ARGUMENTS

The replace_clock_gates command has no arguments.

DESCRIPTION

This command performs replaces all combinational cells in the design that are clock gates with integrated latch-based clock gates. Before clock-gating replacement can be performed, the clock ports must be identified with the **create_clock** command. All combinational cells in the design that are identified as clock gates are candidates for replacement. The combinational cells that can be considered for replacement possess the following **function_id {a2.0, la2.0, a2.1, la2.1, a2.2, la2.2}**. The basic rule for replacement is that any AND cell with function_id a2.0 shall be replaced by a rising-edge integrated latch-based clock gate.

If a combinational cell has two clock different clock inputs it is not considered for replacement.

A combinational cell that has edge conflicts with its registers cannot be replaced. For example, if an AND cell is driving a falling-edge register, it is not replaced. The number of cells with edge conflicts will be printed as an output of the execution.

If a combinational cell has any constraints, they are inherited by the new replacement cell, unless the combinational cell has restrictions that void replacement. When a cell is reported as restricted it could be due to an inferred or an user restriction. The number of restricted cells will be printed as an output of the execution.

If a design has already been placed, no replacement is carried out in the entire design.

Multicorner-multimode is not supported. The replacement occurs in all scenarios defined in the session.

If a combinational cell drives a register that has already been scan-stitched, no replacement is performed for that combinational cell. The number of scan-stitched cells will be printed as an output of the execution.

Make sure that **set verification_clock_gate_hold_mode low** is set on **Formality** configuration.

EXAMPLES

The commands in the following example show the typical flow for using the **replace_clock_gates** command. After reading or elaborating the design and defining the clock ports, the **replace_clock_gates** command replaces the combinational clock gates.

```
prompt> read_verilog design.v

prompt> current_design top

prompt> link

prompt> create_clock clk

prompt> replace_clock_gates
 Combinational clock tree candidates 8
 Replaced 4
 Not Replaced 4
   Scan stitched 0
   Edge conflict 0
   Restricted 4
```

SEE ALSO

compile(2)
report_cells(2)
report_clock_gating(2)

replace_fillers_by_rules

Replaces fillers based on specified replacement rules.

SYNTAX

status **replace_fillers_by_rules**
-replacement_rule *rule*
[-constraint_fillers *cons_lib_cells*]
[-non_constraint_fillers *non_cons_lib_cells*]
[-non_constraint_left_fillers *non_cons_left_lib_cells*]
[-non_constraint_right_fillers *non_cons_right_lib_cells*]
[-no_violation_along_exception_cells]
[-max_constraint_length *distance*]
[-exception_cells *exception_lib_cells*]
[-replace_abutment *target_custom_lib_cell_pair*]
[-illegal_abutment *lib_cells*]
[-tap_cells *lib_cells*]
[-adjacent_non_od_cells *lib_cells*]
[-left_violation_tap *lib_cell*]
[-right_violation_tap *lib_cell*]
[-both_violation_tap *lib_cell*]
[-tap_distance_range *min_max*]
[-prefix *prefix*]
[-target_fillers *exception_lib_cells*]
[-check_only]
[-left_end *orientation_type*]
[-right_end *orientation_type*]
[-refill_table *table*]
[-random_replace *table*]
[-layer *layer_name*]
[-n_left *lib_cells*]
[-n_right *lib_cells*]
[-n_left_right *lib_cells*]
[-n_left_center_right *lib_cells*]
[-n_none *lib_cells*]
[-p_left *lib_cells*]
[-p_right *lib_cells*]
[-p_left_right *lib_cells*]
[-p_left_center_right *lib_cells*]
[-p_none *lib_cells*]
[-small_fillers *lib_cells*]
[-replacement_fillers *lib_cells*]

Data Types

rule string

cons_lib_cells collection
non_cons_lib_cells collection
non_cons_left_lib_cells collection
non_cons_right_lib_cells collection
distance float
exception_lib_cells collection
target_custom_lib_cell_pair collection
lib_cells collection
lib_cell string
min_max pair of floats
prefix string
exception_lib_cells collection
orientation_type string
table list of list
layer_name string

ARGUMENTS

-replacement_rule *rule*

Specify which rule you want to apply. You must specify one of the following keywords:

- **max_vertical_constraint**
  Repair max vertical edge length violations: It is a violation when a column of vertically aligned and constrained cells has a total stacked height greater than max_constraint_length. It is repaired by replacing fillers from the constraint list with the ones from the non_constraint list. Options for this rule are: **-constraint_fillers**, **-non_constraint_fillers**, **-exception_cells**, **-max_constraint_length**, **-non_constraint_left_fillers**, **-non_constraint_right_fillers** and **-no_violation_along_exception_cells**.

- **half_row_adjacency**
  This replacement rule scans for half height fillers and replace them with correct half height fillers based on if abutting cells are inbound cells or regular cells. The replacement half height fillers are specified in following options:-n_left, -n_right, -p_left, -p_right, -n_left_right, -p_left_right, -n_left_center_right, -p_center_right, -n_none, -p_none. See the description for each option for details

- **boundary_cell_vertical_constraint**
  This replacement rule scans the vertical stack distance of the left and right boundary cells specified by left_bdy_cell and right_bdy_cell. If the vertical stack exceed the distance specified, the boundary cell at the location will be replaced by left_replacement_cell or right_replacement_cell. Cells specified in the exception cell list (exception_cells) are considered "safe" and will cause the vertical scanning to be reset. A boundary cell not specified on the left/right/exception cell list are considered "unsafe" and will be included in vertical stack distance but will not be replaced. The replacement cell must have the same height as the boundary cells specified. Only left/right boundary cells are replaced. Horizontal boundary cells are not processed. Options for this rule are: **-left_boundary_cell**, **-right_boundary_cell**, **-exception_cells**, **-max_constraint_length**, **-left_replacement_cell**, **-right_repelacement_cell**

- **illegal_abutment**
  Repair illegal abutment violations: It is a violation if a target filler abuts a filler of the same library cell or cells in the illegal_abutment list. This violation is repaired by replacing the target filler with matching custom filler. Options for this rule are: **-replace_abutment** and **-illegal_abutment**.

- **od_tap_distance**
  Repair Tap OD width violations: It is a violation when OD layer distance between tap cell and neighboring cells falls within tap_distance_range. If **-adjacent_non_od_cells** are specified, these cells are assumed to have no OD layer and OD distance is measured to the nearest cells with OD. If **-adjacent_non_od_cells** is not specified, the OD distance is measured

between tap cell and abutting neighbors. This violation is repaired by replacing tap cells with left_vilation_tap, right_violation_tap, or both_violation_tap if violation is on the left, right or both side. Options for this rule are: **-tap_cells**, **-adjacent_non_od_cells**, **-left_violation_tap**, **-right_violation_tap**, **-both_violation_tap**, and **-tap_distance_range**.

- **od_horizontal_distance**
  Max horizontal length violations: It is a violation when a continuous row of constrained cells has a total horizontal length greater than max_constraint_length. This violation is repaired by replacing the target filler with a matching custom filler. Options for this rule are: **-max_constraint_length** and either specify replace rule using table style with **-refill_table** or by using list style with **-constraint_fillers**, **-non_constraint_fillers**, or **-exception_cells**

  Note that command does not check for any other rules, so when your design has multiple custom fillers of same size, you should use **-refill_table** to provide a map for the tool when replacing target fillers with other custom fillers. For example, when fillers have min-vth width rules, you should use **-refill_table** to specify a custom filler to use to replace target fillers of same vt to avoid introducing new vt layer rule violations

- **random**
  Replace specified fillers with another fillers randomly chosen from a list. Option for this rule is: **-random_replace**

- **horizontal_edges_distance**
  This rule will scan for a continuous horizontal sequence of abutted layer objects for a maximum of the given distance. When a violator cell in the constraint list is found in the current row, replace it with a non-constraint filler cell. Option for this rule is: **-constraint_fillers, -non_constraint_fillers, -layer, -max_constraint_length, -refill_table**

- **small_filler_stacking**
  Repair max vertical small filler stacking violations: This type of violation occurs when vertically stacked small fillers (no greater than 2x) excceds more than max_constraint_length. Two ways to repair this violation: replace a series of horizontally abuting small fillers with a replacement filler, or swap a small filler with horizontally abutting replacement filler. Tool will use replacement fillers of same VT type as small fillers. Tool however, will not check any other rule. Options for this rule are: **-small_fillers**, **-replacement_fillers**, **-max_constraint_length**.

- **end_orientation**
  Modify the orientation of the left-most and/or right-most filler cells in a continuous horizontal sequence of specified filler cells so that they have the correct orientation. The continuous sequence of cells must have at least two filler cells. Options for this rule are: **-target_fillers**, **-left_end**, **-right_end**, and **-check_only**.

-small_fillers *lib_cells*

To be used with small_filler_stacking rule. Use this option to specify small fillers (no greater than 2x).

-replacement_fillers *lib_cells*

To be used with small_filler_stacking rule. Use this option to specify replacement fillers (greater than 2x).

-n_left *lib_cells*

To be used with half_row_adjacency rule. Use fillers from this list for n-type half height filler abutting inbound cells on left side only.

-n_right *lib_cells*

To be used with half_row_adjacency rule. Use fillers from this list for n-type half height filler abutting inbound cells on right side only.

-n_left_right *lib_cells*

To be used with half_row_adjacency rule. Use fillers from this list for n-type half height filler abutting inbound cells on left and right sides only.

-n_left_center_right *lib_cells*

To be used with half_row_adjacency rule. Use fillers from this list for n-type half height filler abutting inbound cells on right, left

and center sides only.

-n_none ***lib_cells***

To be used with half_row_adjacency rule. Use fillers from this list for n-type half height filler not abutting inbound cells.

-p_left ***lib_cells***

To be used with half_row_adjacency rule. Use fillers from this list for p-type half height filler abutting inbound cells on left side only.

-p_right ***lib_cells***

To be used with half_row_adjacency rule. Use fillers from this list for p-type half height filler abutting inbound cells on right side only.

-p_left_right ***lib_cells***

To be used with half_row_adjacency rule. Use fillers from this list for p-type half height filler abutting inbound cells on left and right sides only.

-p_left_center_right ***lib_cells***

To be used with half_row_adjacency rule. Use fillers from this list for p-type half height filler abutting inbound cells on right, left and center sides only.

-p_none ***lib_cells***

To be used with half_row_adjacency rule. Use fillers from this list for p-type half height filler not abutting inbound cells.

-left_boundary_cell ***lib_cell***

Specifies the left boundary cell that are to be considered for replacement

-right_boundary_cell ***lib_cell***

Specifies the right boundary cell that are to be considered for replacement

-left_replacement_cell ***lib_cell***

Specifies the lib cell that will be used to replace -left_boundary_cell

-right_replacement_cell ***lib_cell***

Specifies the lib cell that will be used to replace -right_boundary_cell

-layer ***layer_name***

Specify the layer name.

-random_replace ***table***

Specify a list of {target {custom cell list}} library cell pairs. Target filler cell will be replaced by a filler randomly chosen from custom cell list. Note that not all target fillers are replaced. The goal is to have target + custom cells in the design equally. Target and custom cells must be of same dimension.

-replace_abutment ***target_custom_lib_cell_pair***

Specify a list of {target custom} library cell pairs. Target filler cell will be replaced by pairing custom filler cell when it has illegal abutment. Please specify full lib cell name. Patten matching (*) cannot be used

-illegal_abutment ***lib_cells***

Specify the list of library cells that are considered illegal abutment when abutting to one.

-constraint_fillers *cons_lib_cells*

Specifies the list of library filler cells that can be replaced, if needed.

-non_constraint_fillers *non_cons_lib_cells*

Specifies the list of library filler cells that can used to replace the constrained ones.

-non_constraint_left_fillers *non_cons_left_lib_cells*

Specifies the list of library filler cells that have property that the left side of the cell can be used to break a continuous edge of constrained cells.

-non_constraint_right_fillers *non_cons_right_lib_cells*

Specifies the list of library filler cells that have property that the right side of the cell can be used to break a continuous edge of constrained cells.

-no_violation_along_exception_cells

When this option is specified, the edges of cells abutting exception cells are not considered to be violating the maximum constraint length.

-max_constraint_length *distance*

Specifies the maximum length of constrained cells that can align vertically, or can be continuously horizontally without a custom filler or exception cell.

-exception_cells *exception_lib_cells*

Specifies the list of library cells (typically not fillers) that are to be considered unconstrained. All cells that are not listed are considered constrained.

-target_fillers *exception_lib_cells*

Specifies the list of library cells that are considered for continuous horizontal sequence checking for the end_orientation replacement rule.

-check_only

When this option is specified the command reports violations rather than fixing them. This option only works with the end_orientation rule replacement rule.

-left_end *orientation_type*

Specifies how to correct the orientation of the left-most filler cell in the continuous horizontal sequence of filler cells. The following strings can be specified: "inverse", "R0_or_MX" or "MY_or_R180". If R0_or_MX is specified, the orientation of the cell is changed to R0 (MX) if it was MY (R180). If MY_or_R180 is specified, the orientation of the cell is changed to MY (R180) if it was R0 (MX).

-right_end *orientation_type*

Specifies how to correct the orientation of the right-most filler cell in the continuous horizontal sequence of filler cells. The following strings can be specified: "inverse", "R0_or_MX" or "MY_or_R180". If R0_or_MX is specified, the orientation of the cell is changed to R0 (MX) if it was MY (R180). If MY_or_R180 is specified, the orientation of the cell is changed to MY (R180) if it was R0 (MX).

-refill_table *table*

Specify replacement rule using table style. The syntax is:

-refill_table {{{ customA } { targetA1 targetA2... }}
{ customB } { targetB1 targetB2 ...}}}

customA is replaces targetA1 and targetA2, while customB replaces targetB1 and targetB2. Please specify full lib cell name. Patten matching (*) cannot be used

-prefix *prefix*

Adds the specified prefix to the instance names of new filler cells inserted by this command. You can later use the prefix to identify the new filler cell instances.

-left_violation_tap

This option is for doing Tap OD width violations fix. Use this option to specify replacement tap to fix violations on left. See "-rules {od_tap_distance} " for details.

-right_violation_tap

This option is for doing Tap OD width violations fix. Use this option to specify replacement tap to fix violations on right. See "-rules {od_tap_distance} " for details.

-both_violation_tap

This option is for doing Tap OD width violations fix. Use this option to specify replacement tap to fix violations on both sides. See "-rules {od_tap_distance} " for details.

-tap_cells

This option is for doing Tap OD width violations fix. Use this option to specify existing tap cells to be checked. See "-rules {od_tap_distance} " for details.

-tap_distance_range

This option is for doing Tap OD width violations fix. Specify tap OD distance range. It is a violation if tap OD distnace falls inside this range . See "-rules {od_tap_distance} " for details.

-adjacent_non_od_cells

This option is for doing Tap OD width violations fix. By default, OD distance is measured between tap cell and cells abutting the tap cell. This option specify list of cells that have no OD. When tap is adjacent such non-OD cell, those NO-OD cells skipped and the distance is measured against first cell that is not a non-od cell. See "-rules {od_tap_distance} " for details.

-skip_fixed_cells

Skips fixed and locked cells during filler cell replacement. This option is supported only for the following rules: **illegal_abutment**, and **random**. The command fails if you use this option with an unsupported rule.

DESCRIPTION

This command repairs various placement rule violations by replacing target fillers with custom fillers. It does not repair standard cell placement errors. You must specify which rule to apply by specifying the **-replacement_rule** option. The supported rules are: max vertical edge length rule, FILL1 illegal abutments, tap OD width violations, and max horizontal length rule.

1. Max vertical edge length violations: This type of violations occurs when a column of vertically aligned constrained cells has a total stacked height greater than max_constraint_length. It is repaired by replacing fillers from the constraint list with the ones from the non_constraint list. Options for this rule are: **-constraint_fillers**, **-non_constraint_fillers**, **-exception_cells**, and **-max_constraint_length**.

2. FILL1 illegal abutment violations: It is a violation if a target filler abuts filler of same library cell or cells in illegal_abutment list. This is repaired by replacing target filler with matching custom filler. Options for this rule are: **-replace_abutment** and **-illegal_abutment**.

3. Tap OD width violations: It is a violation when total OD width of a tap cell and abutting cell falls within tap_distance_range. It is repaired by replacing tap cells with left_vilation_tap, right_violation_tap, or both_violation_tap if violation is on the left, right or both side. Options for this rule are: **-tap_cells**, **-left_violation_tap**, **-right_violation_tap**, **-both_violation_tap**, and **-tap_distance_range**.

4. Max horizontal length violations: It is a violation when a continuous row of constrained cells has a total horizontal length greater than max_constraint_length. It is repaired by replacing target filler with matching custom filler. Options for this rule are: **-constraint_fillers**, **-non_constraint_fillers**, **-exception_cells**, and **-max_constraint_length**.

5. Random replace: This replace a target filler randomly chosen from a list of other fillers. Note that not all target fillers are replaced. The goal is to have target + custom cells in the design equally, but it is not possible because filler insertion with **create_stdcell_fillers** always use the first legal filler listed. So during filler insertion, you specify only target fillers, and then use **replace_fillers_by_rules -replacement_rule random** to randomly replace target fillers with custom fillers.

The command is meant to be run after adding filler cells. It can only repair one rule at a time. It does not consider other placement rule other than the rule it is repairing. It is your responsibility to ensure that filler swapping with **replace_fillers_by_rules** will not create other rule violations.

EXAMPLES

The following example repairs max vertical edge length violations.

```
prompt> replace_fillers_by_rules \
    -replacement_rule {max_vertical_constraint} \
    -constraint_fillers  [get_lib_cells  { mylib/FILL_4X mylib/FILL_2X } ] \
    -non_constraint_fillers [get_lib_cells { mylib/FILL_1X } ] \
    -max_constraint_length 50 \
    -exception_cells [get_lib_cells { mylib/BUFF_1 } ]
```

The following example repairs max boundary cell vertical stack length violations.

```
prompt> replace_fillers_by_rules \
    -replacement_rule {boundary_cell_vertical_constraint} \
    -left_boundary_cell {old_left_cell} \
    -left_replacement_cell { new_left_cell} \
    -right_boundary_cell {old_right_cell} \
    -right_replacement_cell { new_right_cell} \
    -max_constraint_length 50 \
    -exception_cells [get_lib_cells { mylib/other_boundary_cell } ]
```

The following example repairs max small filler stacking length violations.

```
prompt> replace_fillers_by_rules \
    -replacement_rule {small_filler_stacking} \
    -small_fillers {FILL1SVT FILL1LVT FILL2SVT FILL2LVT} \
    -replacement_fillers { FILL3SVT FILL3LVT FILL4SVT FILL4LVT} \
    -max_constraint_length 20
```

The following example repairs FILL1 abutment violations:

```
prompt> replace_fillers_by_rules \
    -replacement_rule {illegal_abutment } \
    -replace_abutment  { {target_FILL1a custom_FILL1a} { target_FILL1b custom_FILL1b} } \
```

```
    -illegal_abutment [get_lib_cells { mylib/FILL_2* } ]
```

The following example repairs Tap OD width violations:

```
prompt> replace_fillers_by_rules \
    -replacement_rule { od_tap_distance } \
    -tap_cells  { mylib/TARGET_TAP } \
    -left_violation_tap {mylib/TAP_L } \
    -right_violation_tap {mylib/TAP_R}  \
    -both_violation_tap { mylib/TAP_BOTH}
```

The following example repairs max horizontal length violations using list style. It will use FILL_1X to replace FILL_4X or FILL_2X:

```
prompt> replace_fillers_by_rules \
    -replacement_rule { od_horizontal_distance } \
    -constraint_fillers  [get_lib_cells  { mylib/FILL_4X mylib/FILL_2X } ] \
    -non_constraint_fillers [get_lib_cells { mylib/FILL_1X } ] \
    -max_constraint_length 50 \
    -exception_cells [get_lib_cells { mylib/BUFF_1 } ]
```

The following example repairs max horizontal length violations using table style. customA is used to replace targetA1 and targetA2, while customB is used to replace targetB1 and targetB2:

```
prompt> replace_fillers_by_rules \
    -replacement_rule { od_horizontal_distance } \
    -refill_table { { {customA} { targetA1 targetA2} }
            { {customB} { targetB1 targetB2} }}
    -max_constraint_length 50
```

The following example randomly replace FILL4M with FILL4MTH or FILL4MTL, and replace FILL2M with FILL2MTH or FILL2MTL:

```
prompt> replace_fillers_by_rules \
     -replacement_rule { random } \
     -random_replace {{{FILL4M} {FILL4MTH FILL4MTL}} \
             {{FILL2M} {FILL2MTH FILL2MTL}}}
```

SEE ALSO

create_stdcell_fillers(2)
create_tap_cells(2)

report_3d_chip_placement

Reports placement-related information of the chip in a 3DIC top-level design.

SYNTAX

status **report_3d_chip_placement**
-chips *chips*
-all

Data Types

chips list or collection

ARGUMENTS

-chips *chips*

Specifies a list of chip names to report. In a 3DIC design, the chip names are the cell instance names.

-all

Reports all chips within the 3DIC design.

DESCRIPTION

This command reports placement information set by the **set_cell_location** command for a 3DIC design. In a 3DIC design, multiple dies are stacked vertically or placed side-by-side on one large silicon interposer, and the collection of chips is modeled as a virtual top-level design. The individual chips are modeled as cells within the top-level design.

The placement information includes the design origin which describes the origin point of the chip relative to the 3-dimensional coordinate system. The orientation determines how the chip rotates in 3D space using the 3-dimensional coordinate system. The scaling factor describes the actual size of chip in the design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example command reports information for a chip named **cpu**.

```
prompt> set_cell_location cpu -coordinates {300 400 1} -orientation FS -z_offset 1
1
prompt> report_3d_chip_placement -chips cpu
--------------------------------------------------------------------------------
chip_name      stack_z     location          orientation   scaling_factor
--------------------------------------------------------------------------------
cpu              1    (300.0000  400.0000 )     FS         1.000000
```

SEE ALSO

check_3d_design(2)
set_cell_location(2)

report_abstracts

Reports details of abstract views for the specified designs.

SYNTAX

status **report_abstracts**
[*design_list*]

Data Types

design_list list

ARGUMENTS

design_list

Specifies the list of designs for which to report abstract views. By default, the command generates a report for all the abstract view designs linked in the current design.

DESCRIPTION

Reports details of abstract view designs. If one or more abstract view designs are given, the command generates a report for the specified designs. If no abstract view design is given, the command generates a report for all the abstract view designs linked in the current design, as well as report the top level compression data.

EXAMPLES

The following example reports details about the abstract view of design BlkA:

prompt> **report_abstracts [get_designs BlkA -filter "view_name == abstract"]**

The following example reports all the abstract view designs linked in the current design, plus the compression data at the current design level.

prompt> **report_abstracts**

SEE ALSO

create_abstract(2)

report_activity

Reports switching activity.

SYNTAX

int **report_activity**
[-rtl]
[-driver]
[-verbose]
[-show_zeros]
[-scenarios *scenario_list*]
[-modes *mode_list*]
[-corners *corner_list*]
[-print_objects {*activity_type_list object_type_list*}]

Data Types

scenario_list list
mode_list list
corner_list list
activity_type_list list
object_type_list list

ARGUMENTS

-rtl

Reports switching activity similar to the **report_saif -rtl_saif** command in Design Compiler. The object types listed are port, sequential cell ("seq-cell") and tristate cells ("tri-cell").

-driver

Reports drivers based on their logic function. Two special situations occur: 1) a driver with an unknown function ("no-func") and 2) signals that have no driver at all ("no-driver"). The logic function listed are: primary input ports ("primary-input"), primary inout ports ("primary-inout"), sequential pins ("seq-pin"), tristate pins ("tri-pin"), combinational pins ("comb-pin"), unknown function ("no-func"), and no drivers ("no-driver"). If user essential activity points have been created, these will be printed in a second report as well.

-verbose

Reports minor switching activity categories in addition to the default major categories. By default, the report shows the major categories for activity types: "simulated" (retrieved from simulation files), "annotated" (annotated by the user), "derived" (activities retrieved from constraints), "calculated" (activities computed by activity propagation) and "default" (all other activities). If this option is used also the minor categories are shown.

For "simulated" there are no minor categories.

For "annotated" the minor categories are "set_switching_activity" (set by the user) and "abstract" (created during abstraction)

For "derived" the minor categories are "create_clock" (from SDC), "create_generated_clock" (from SDC), "set_case_analysis" (from SDC), "timer_implied" (propagated set_case_analysis by the timer), "logic_constant" (from netlist) and "supply_constant" (from netlist).

For "calculated" the minor categories are "implied" (propagation of activity through repeater cells), "propagated" (computed after propagate_switching_activity) and estimated (created during optimization).

For "default" there are no minor categories.

-show_zeros

Include zero-valued columns in the report. By default, zero-valued columns are suppressed in the report.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Output of the command is generated separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command reports for the current scenario in the design.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering on modes occurs.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If this option is not specified, no filtering occurs on corners.

-print_objects {*activity_type_list object_type_list*}

Prints all objects counted in the report with the given set of activity_types and object_types. Use the shorthand "all" for all activity types or all object types.

DESCRIPTION

Reports the switching activity type for objects. The objects that are reported depends on the command options used. By default, this command mimics the **report_saif** command in Design Compiler. With the **-rtl** option this command mimics the **report_saif -rtl_saif** command in Design Compiler. With the **-driver** option this command prints out a driver-centric report.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify different scenarios, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following examples report switching activities for the current design.

```
prompt> report_activity
****************************************
Report : report_activity
Design : ChipTop
Version: M-2016.12
Date   : Tue Dec  6 04:48:13 2016
****************************************

Scenario 'default' (mode 'default', corner 'default')

Activity Type            port          net       leaf-pin
        block-pin      hier-pin     cell-SDPD
-------------------------------------------------------------------------
simulated          0 (  0.0%)    0 (  0.0%)     0 (  0.0%)
       0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
annotated          0 (  0.0%)    0 (  0.0%)     0 (  0.0%)
       0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
derived            1 (  1.4%)    9 (  0.1%)  13076 ( 36.9%)
       0 (  0.0%)   75 (  6.4%)    0 (  0.0%)
calculated        33 ( 47.8%)  6160 ( 95.0%)  22299 ( 62.8%)
       0 (  0.0%)  1002 ( 86.0%)    0 (  0.0%)
default           35 ( 50.7%)   312 (  4.8%)    109 (  0.3%)
       0 (  0.0%)   88 (  7.6%)    0 (  0.0%)
-------------------------------------------------------------------------
total             69 (100.0%)  6481 (100.0%)  35484 (100.0%)
       0 (  0.0%)  1165 (100.0%)    0 (100.0%)

prompt> report_activity -verbose
****************************************
Report : report_activity
       -verbose
Design : ChipTop
Version: M-2016.12
Date   : Tue Dec  6 06:43:56 2016
****************************************

Scenario 'default' (mode 'default', corner 'default')

Activity Type                  port          net       leaf-pin
        block-pin      hier-pin     cell-SDPD
-------------------------------------------------------------------------
simulated                0 (  0.0%)    0 (  0.0%)     0 (  0.0%)
       0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
annotated
  set_switching_activity  0 (  0.0%)    0 (  0.0%)     0 (  0.0%)
       0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
  abstract                0 (  0.0%)    0 (  0.0%)     0 (  0.0%)
       0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
derived
  create_clock            1 (  1.4%)    1 (  0.0%)    79 (  0.2%)
       0 (  0.0%)    3 (  0.3%)    0 (  0.0%)
  create_generated_clock  0 (  0.0%)    0 (  0.0%)     0 (  0.0%)
       0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
  set_case_analysis       0 (  0.0%)    0 (  0.0%)     0 (  0.0%)
       0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
  timer_implied           0 (  0.0%)    0 (  0.0%)     0 (  0.0%)
```

```
        0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
  logic_constant         0 (  0.0%)    3 (  0.0%)     0 (  0.0%)
        0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
  supply_constant        0 (  0.0%)    5 (  0.1%)  12997 ( 36.6%)
        0 (  0.0%)   72 (  6.2%)    0 (  0.0%)
calculated
  implied               0 (  0.0%)   46 (  0.7%)    701 (  2.0%)
        0 (  0.0%)   45 (  3.9%)    0 (  0.0%)
  propagated           33 ( 47.8%)  6114 ( 94.3%)  21598 ( 60.9%)
        0 (  0.0%)  957 ( 82.1%)    0 (  0.0%)
  estimated             0 (  0.0%)    0 (  0.0%)     0 (  0.0%)
        0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
default                35 ( 50.7%)   312 (  4.8%)    109 (  0.3%)
        0 (  0.0%)   88 (  7.6%)    0 (  0.0%)
-------------------------------------------------------------------------
total                  69 (100.0%)  6481 (100.0%)  35484 (100.0%)
        0 (  0.0%)  1165 (100.0%)    0 (100.0%)

prompt> report_activity \
  -print_object {{derived simulated} {port net}} -verbose
****************************************
Report : report_activity
       -verbose
Design : ChipTop
Version: M-2016.12
Date   : Tue Dec  6 06:36:20 2016
****************************************

Scenario 'default' (mode 'default', corner 'default')

Activity Type                  port          net        leaf-pin
        block-pin      hier-pin     cell-SDPD
-------------------------------------------------------------------------
simulated              0 (  0.0%)    0 (  0.0%)      0 (  0.0%)
        0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
annotated
  set_switching_activity  0 (  0.0%)    0 (  0.0%)      0 (  0.0%)
        0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
  abstract               0 (  0.0%)    0 (  0.0%)      0 (  0.0%)
        0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
derived
  create_clock           1 (  1.4%)    1 (  0.0%)     79 (  0.2%)
        0 (  0.0%)    3 (  0.3%)    0 (  0.0%)
  create_generated_clock  0 (  0.0%)    0 (  0.0%)      0 (  0.0%)
        0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
  set_case_analysis      0 (  0.0%)    0 (  0.0%)      0 (  0.0%)
        0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
  timer_implied          0 (  0.0%)    0 (  0.0%)      0 (  0.0%)
        0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
  logic_constant         0 (  0.0%)    3 (  0.0%)      0 (  0.0%)
        0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
  supply_constant        0 (  0.0%)    5 (  0.1%)   12997 ( 36.6%)
        0 (  0.0%)   72 (  6.2%)    0 (  0.0%)
calculated
  implied               0 (  0.0%)   46 (  0.7%)    701 (  2.0%)
        0 (  0.0%)   45 (  3.9%)    0 (  0.0%)
```

```
 propagated          33 ( 47.8%)  6114 ( 94.3%)   21598 ( 60.9%)
      0 (  0.0%)   957 ( 82.1%)     0 (  0.0%)
 estimated            0 (  0.0%)    0 (  0.0%)      0 (  0.0%)
      0 (  0.0%)    0 (  0.0%)     0 (  0.0%)
default              35 ( 50.7%)   312 (  4.8%)    109 (  0.3%)
      0 (  0.0%)    88 (  7.6%)     0 (  0.0%)
-------------------------------------------------------------------------
total                69 (100.0%)  6481 (100.0%)   35484 (100.0%)
      0 (  0.0%)  1165 (100.0%)     0 (100.0%)

List of requested objects:
create_clock    port clock
logic_constant  net  GPRs/*Logic0*
logic_constant  net  InstDecode/*Logic0*
logic_constant  net  Multiplier/*Logic0*
supply_constant net  VDD108
supply_constant net  VDD108_SD
supply_constant net  VDD90
supply_constant net  VDD90_SD
supply_constant net  VSS

prompt> report_activity -rtl
****************************************
Report : report_activity
      -rtl
Design : ChipTop
Version: M-2016.12
Date   : Tue Dec  6 04:54:26 2016
****************************************

Scenario 'default' (mode 'default', corner 'default')

Object Type         derived      calculated         default     total
-------------------------------------------------------------------------
port            1 (  1.4%)     33 ( 47.8%)     35 ( 50.7%)       69
seq-cell        0 (  0.0%)    677 (100.0%)      0 (  0.0%)      677
tri-cell        0 (  0.0%)      0 (  0.0%)      0 (  0.0%)        0

Unlisted columns (all zeros):
   simulated
   annotated

prompt> report_activity -rtl -verbose
****************************************
Report : report_activity
      -rtl
      -verbose
Design : ChipTop
Version: M-2016.12
Date   : Tue Dec  6 06:45:41 2016
****************************************

Scenario 'default' (mode 'default', corner 'default')

Object Type     create_clock      propagated         default     total
-------------------------------------------------------------------------
port            1 (  1.4%)     33 ( 47.8%)     35 ( 50.7%)       69
```

```
seq-cell         0 (  0.0%)     677 (100.0%)      0 (  0.0%)      677
tri-cell         0 (  0.0%)       0 (  0.0%)      0 (  0.0%)        0

Unlisted columns (all zeros):
   simulated
   set_switching_act
   abstract
   create_gen_clock
   set_case_analysis
   timer_implied
   logic_constant
   supply_constant
   implied
   estimated

prompt> report_activity -driver
****************************************
Report : report_activity
       -driver
Design : ChipTop
Version: M-2016.12-VAL
Date   : Tue Dec  6 04:55:41 2016
****************************************

Scenario 'default' (mode 'default', corner 'default')

Activity Type    primary-input         seq-pin         comb-pin
       no-driver     total
-------------------------------------------------------------------
simulated          0 (  0.0%)       0 (  0.0%)      0 (  0.0%)
      0 (  0.0%)         0
annotated          0 (  0.0%)       0 (  0.0%)      0 (  0.0%)
      0 (  0.0%)         0
derived            1 (  2.8%)       0 (  0.0%)      0 (  0.0%)
      8 (  2.8%)         9
calculated         0 (  0.0%)     709 (100.0%)    5451 (100.0%)
      0 (  0.0%)      6160
default           35 ( 97.2%)       0 (  0.0%)      0 (  0.0%)
    277 ( 97.2%)       312
-------------------------------------------------------------------
total             36 (100.0%)     709 (100.0%)    5451 (100.0%)
    285 (100.0%)      6481

Unlisted columns (all zeros):
   primary-inout
   tri-pin
   no-func

Activity Type     my_essential    essential_too     total
-----------------------------------------------------------
simulated          0 (  0.0%)       0 (  0.0%)         0
annotated          0 (  0.0%)       0 (  0.0%)         0
derived            0 (  0.0%)       0 (  0.0%)         0
calculated         1 (100.0%)       1 (100.0%)         2
default            0 (  0.0%)       0 (  0.0%)         0
-----------------------------------------------------------
```

```
total            1 (100.0%)      1 (100.0%)       2

prompt> report_activity -driver -verbose
****************************************
Report : report_activity
       -driver
       -verbose
Design : ChipTop
Version: M-2016.12
Date   : Tue Dec  6 06:46:29 2016
****************************************

Scenario 'default' (mode 'default', corner 'default')

Activity Type          primary-input      seq-pin      comb-pin
       no-driver total
------------------------------------------------------------------------
simulated                 0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
     0 (  0.0%)    0
annotated
 set_switching_activity   0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
     0 (  0.0%)    0
 abstract                 0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
     0 (  0.0%)    0
derived
 create_clock             1 (  2.8%)    0 (  0.0%)    0 (  0.0%)
     0 (  0.0%)    1
 create_generated_clock   0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
     0 (  0.0%)    0
 set_case_analysis        0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
     0 (  0.0%)    0
 timer_implied            0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
     0 (  0.0%)    0
 logic_constant           0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
     3 (  1.1%)    3
 supply_constant          0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
     5 (  1.8%)    5
calculated
 implied                  0 (  0.0%)    0 (  0.0%)   46 (  0.8%)
     0 (  0.0%)   46
 propagated               0 (  0.0%)  709 (100.0%)  5405 ( 99.2%)
     0 (  0.0%)  6114
 estimated                0 (  0.0%)    0 (  0.0%)    0 (  0.0%)
     0 (  0.0%)    0
default                  35 ( 97.2%)    0 (  0.0%)    0 (  0.0%)
    277 ( 97.2%)  312
------------------------------------------------------------------------
total                    36 (100.0%)  709 (100.0%)  5451 (100.0%)
    285 (100.0%)  6481

Unlisted columns (all zeros):
   primary-inout
   tri-pin
   no-func

Activity Type              my_essential    essential_too     total
```

```
-------------------------------------------------------------------------
simulated                 0 (  0.0%)      0 (  0.0%)       0
annotated
  set_switching_activity  0 (  0.0%)      0 (  0.0%)       0
  abstract                0 (  0.0%)      0 (  0.0%)       0
derived
  create_clock            0 (  0.0%)      0 (  0.0%)       0
  create_generated_clock  0 (  0.0%)      0 (  0.0%)       0
  set_case_analysis       0 (  0.0%)      0 (  0.0%)       0
  timer_implied           0 (  0.0%)      0 (  0.0%)       0
  logic_constant          0 (  0.0%)      0 (  0.0%)       0
  supply_constant         0 (  0.0%)      0 (  0.0%)       0
calculated
  implied                 0 (  0.0%)      0 (  0.0%)       0
  propagated              1 (100.0%)      1 (100.0%)       2
  estimated               0 (  0.0%)      0 (  0.0%)       0
default                   0 (  0.0%)      0 (  0.0%)       0
-------------------------------------------------------------------------
total                     1 (100.0%)      1 (100.0%)       2
```

SEE ALSO

get_switching_activity(2)
propagate_switching_activity(2)
read_saif(2)
reset_switching_activity(2)
set_switching_activity(2)

report_annotated_check

Displays all annotated timing checks on the current design.

SYNTAX

status **report_annotated_check**
[-nosplit]

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

This command lists all annotated timing checks in the current design. The pin-to-pin timing checks reported are setup and hold.

To list annotated delays, use the **report_annotated_delay** command.

Multicorner-Multimode Support

This command works only on the current scenario.

EXAMPLES

The following is an example of an annotated report:

```
prompt> report_annotated_check

****************************************
Report : annotated_check
Design : counter
Version: v3.1
Date   : Tue Apr 14 19:42:25 2016
```

```
****************************************

Cell Name     From    To        Rise      Fall      Timing Check
--------------------------------------------------------------------------------
U1            c       d         -0.20     -0.20     hold
U1            c       d          2.20      2.20     setup
```

SEE ALSO

read_timing(2)
remove_annotated_check(2)
report_annotated_delay(2)
reset_design(2)
set_annotated_check(2)

report_annotated_delay

Displays delays annotated on cells and nets of the current design.

SYNTAX

status **report_annotated_delay**
[-cell]
[-net]
[-nosplit]
[-summary]
[-min]

ARGUMENTS

-cell

Reports delay data annotated on cells only. By default, both cell- and net-annotated data are reported.

-net

Reports delay data annotated on nets only. By default, both cell- and net-annotated data are reported.

-nosplit

Prevents line splitting and facilitates writing tools to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-summary

Reports the total number of cell and net delay arcs and how many are annotated with delay data. An arc is considered annotated with delays only if all the four delay values (rise, fall, min, max) are present on it.

-min

Reports minimum delay data annotated on cells and nets of the current design. If you do not use this option, the command reports maximum delay data instead.

DESCRIPTION

The **report_annotated_delay** command lists all delay data annotated on cells and nets in the current design. The cell-annotated delay data consists of pin-to-pin delays. The net-annotated delay data consists of pin-to-pin delays, net capacitance values, and net

resistance values.

Note that the delay values reported are the resulting cell and net delays used in the **report_timing** command and might be different from your annotated values if the delays were annotated using the **-load_delay net** option of the **read_sdf** or **set_annotated_delay** command.

Multicorner-Multimode Support

This command works only on the current scenario.

EXAMPLES

The following is an example of an annotated delay report:

```
prompt> report_annotated_delay

****************************************
Report : annotated -cell
Design : counter
Version: v3.0
Date   : Tue Apr 14 19:42:25 2016
****************************************

Cell Name     From    To       Rise      Fall
-------------------------------------------------------------
CO            A       Z        100.00    100.00

****************************************
Report : annotated -net
Design : counter
Version: v3.0
Date   : Tue Apr 14 19:42:25 2016
****************************************


Net Name      From    To       Rise      Fall     Load      Res.
-------------------------------------------------------------------------
h             ffc/QN  w/A      200.00    200.00    50.00    200.00
h             ffc/QN  m/B      200.00    200.00    50.00    200.00
h             ffc/QN  r/B                 50.00   200.00
m             m/Z     CO/A     200.00    200.00    40.00    100.00
```

SEE ALSO

read_sdf(2)
remove_annotated_delay(2)
reset_design(2)
set_annotated_delay(2)

report_annotated_power

Reports user-annotated cell power dissipation.

SYNTAX

status **report_annotated_power**
-scenarios *scenario_list*
-supply_net *supply_net*
-list_annotated
-significant_digits *digits*

Data Types

scenario_list list
digits integer

ARGUMENTS

-scenarios

Specifies the set of scenarios for which the annotated power should be reported. The default is the current_scenario value.

-supply_net

Specifies the supply net for which the annotated power should be reported. Default, the annotated power of all supply nets is reported.

-list_annotated

Specifies that a per-cell annotation should be printed for all annotated cells. By default, only a summary with the number of annotated cells per cell category is printed.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 to 13. The default is 2.

DESCRIPTION

This command reports on the cells that have user-annotated power values. The annotations were set with the *set_annotated_power* command. The command generates a summary of the number of annotated cells per cell category. Four categories are distinguished.

1. Unresolved black-box cells
2. Black-box cells
3. Physical blocks
4. Leaf cells

Unresolved black-box cells are cells that were not bound to a reference module. Physical blocks are hierarchical cells that are bound to physical block's view.

Multicorner-Multimode Support

EXAMPLES

The following example reports annotated power for all scenarios and all supply nets in the current design. In this design, there is one physical sub block (not annotated). Out of the 36 leaf cells, 1 leaf cell has annotated power.

```
prompt> report_annotated_power

================================================
Scenario: default
================================================

                    |         |         |   NOT   |
   Cell type        |   Total  | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
Physical block          |      1 |      0 |      1 |
Leaf cell               |     36 |      1 |     35 |
-----------------------------+-----------+-----------+-----------+
                    |     37 |      1 |     36 |
```

The following example also reports details about the annotated cells.

```
prompt> report_annotated_power -list_annotated

================================================
Scenario: default
================================================

 Attributes
 ----------
   l - Leaf cell

Annotated cell powers:
-------------------------------
1. bot/c1  (leakage: 0.023) : l


                    |         |         |   NOT   |
   Cell type        |   Total  | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
Physical block          |      1 |      0 |      1 |
Leaf cell               |     36 |      1 |     35 |
-----------------------------+-----------+-----------+-----------+
                    |     37 |      1 |     36 |
```

The following example reports the annotations for supply net VDD only.

```
prompt> report_annotated_power -supply_net VDD

================================================
Supply net: VDD
Scenario: default
================================================

                    |         |         |   NOT   |
   Cell type        |  Total  | Annotated | Annotated |
-----------------------+-----------+-----------+-----------+
Physical block         |      1 |      0 |      1 |
Leaf cell              |     36 |      0 |     36 |
-----------------------+-----------+-----------+-----------+
                    |     37 |      0 |     37 |
```

SEE ALSO

set_annotated_power(2)
remove_annotated_power(2)

report_annotated_transition

Displays annotated transitions on all pins of the current design.

SYNTAX

status **report_annotated_transition**
-nosplit

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_annotated_transition** command lists annotated transition times at every pin in the current design.

Multicorner-Multimode Support

This command works only on the current scenario.

EXAMPLES

The following is an example of an annotated transition time report:

```
prompt> report_annotated_transition

report_annotated_transition

Pin Name    max_rise     max_fall     min_rise     min_fall
------------------------------------------------------------
U0/A        10.00        -            -            17.00
U1/A        -            -            -            9.10
U1/Z        -            -            -            7.00
```

SEE ALSO

read_timing(2)
remove_annotated_transition(2)
reset_design(2)
set_annotated_transition(2)

report_antenna_rules

Name

report_antenna_rules

Reports the given antenna rules defined in the library.

SYNTAX

integer **report_antenna_rules**
[-mode *antenna_mode*]
[-library *library*]
[*rule_names*]

Data Types

antenna_mode integer
library collection
rule_names string

ARGUMENTS

-mode *antenna_mode*

An integer value between 1 and 6, that represents the way the antenna areas are computed. When this option is specified, only the antenna rules that use the given mode for antenna area calculation, are reported. This is an optional option.

-library *library*

Specifies the library from which the antenna rules are to be reported. This is an optional option.

rule_names

Specifies a list of antenna rule-names that are to be reported. This is an optional option. When not specified, all the antenna-rules defined in the current library are reported.

DESCRIPTION

This command reports the given antenna rules for the specified mode that are defined in the given library.

By default, with no options specified, the command reports all the antenna rules defined in the current library. The command returns the number of antenna rules reported.

EXAMPLES

The following example reports all the antenna rule names in the current library that use antenna-mode 1 for antenna area calculation.

```
prompt> report_antenna_rules -mode 1

Rule name  r100
Antenna mode  Diode-mode value
------------  ----------------
 1    1

Column information:
-------------------
   A - Layer name
   B - Layer type
   C - Antenna mode
   D - Ratio
   E - P Ratio
   F - N Ratio
   G - Gate diffusion length
   H - P Gate diffusion length
   I - N Gate diffusion length
   J - Layer scale
   K - Accumulate scale
   L - Protected scale
   M - Diode ratio
   N - Scale factor

   A         B      C      D         E         F         G
   H         I      J      K         L         M         N
---------- ------------ --- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ---------- ---------- ----------
    VIA4      VIA_CUT  1      1.20      1.40      1.80       --
      --        --       --        --        --        --        --

  METAL5 INTERCONNECT  1      2.00      3.10      2.20       --
      --        --       --        --        --        --        --

   METAL INTERCONNECT  1      2.00      3.10      2.20       --
      --        --       --        --        --        --        --

    VIA      VIA_CUT  1      1.20      1.40      1.80       --
      --        --       --        --        --        --        --

  METAL2 INTERCONNECT  1      2.00      3.10      2.20       --
      --        --       --        --        --        --        --
```

```
    VIA2     VIA_CUT  1      1.20      1.40      1.80       --
     --       --       --       --       --       --       --

  METAL3 INTERCONNECT  1      2.00      3.10      2.20       --
     --       --       --       --       --       --       --

    VIA3     VIA_CUT  1      1.20      1.40      1.80       --
     --       --       --       --       --       --       --

  METAL4 INTERCONNECT  1      2.00      3.10      2.20       --
     --       --       --       --       --       --       --

---------- ------------ --- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ---------- ---------- ----------
1
```

SEE ALSO

define_antenna_rule(2)
define_antenna_layer_rule(2)
get_antenna_rule_names(2)
remove_antenna_rules(2)

report_app_options

Reports application options on a block or in the global scope.

SYNTAX

integer **report_app_options**
[-block *block*]
[-global]
[-non_default]
[-as_list]
[-include_equivalent_variable]
[-data *data_file*]
[-csv *file_name*]
[*patterns*]

Data Types

block block name or collection
patterns list
data_file A file-name of an encoded data file. The output of write_app_options.
file_name String. Report contents to this file in CSV format.

ARGUMENTS

-block *block*

Specifies the block, which the options to be reported are set on. This option cannot be specified with the *-global* option. If this option is specified, the command reports the options in the scope of the specified block.

-global

Indicates that options in the global scope will be reported. This option cannot be specified with the *-block* option. This option is deprecated.

-non_default

Returns app options set to different behavior than the system-default. This includes:

- Block-scope app options set to a non-default setting with set_app_options
- Global-scope app options set to a non-default setting with set_app_options
- Block-scope app options given a user default setting with set_app_options -as_user_default

-as_list

Specifies that the report should follow the format expected by the set_app_options command, i.e. one pair of application option {name value} for every line in the report. Selectively reports the application options that has user or system value set.

-include_equivalent_variable

Indicates that the corresponding app var name will be reported for the listed app options. All app options may not have a corresponding app var.

-data *data_file*

A string that represents a filename of a write_app_options *data_file* report_app_options will report app options stored in this file instead of the current executable.

Note: When using this option, obsolete app options are not reported.

-csv *file_name*

Specifies the output CSV file name. The report contents will be output to this file in CSV format. The *file_name* argument can be a local file name or a full path name to the file.

patterns

Specifies a list of patterns for the option names. A pattern can include the wildcard characters asterisk (*) and question mark (?). "*" will match zero or more of any character, and "?" will match any single character. If this option is specified, the command reports those option names that match any of the specified patterns. If not specified, reports all options.

DESCRIPTION

This command generates a report of application options on the specified block or in the global scope, depending on the scope of the application option.

Each block will store only the options explicitly set on it. Any option values not set on the block will be reported as '--' (empty).

The default behavior is to report both the design scoped and global scoped application options, where the value reported for the design scoped application option would be w.r.t the current block.

The report also contains entry for the user-default and system-default.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports application options, both design scoped and global scoped. The design scoped values are w.r.t the current block.

```
prompt> report_app_options
****************************************
Report : app_option
Design : NO_DESIGN
Version: O-2018.06-SP4-BETA
```

```
Date   : Tue Oct 30 19:17:03 2018
****************************************

Name                          Type      Value     User-default System-default Scope     Status    Source
------------------------------------ ---------- ---------- ------------ -------------- ---------- ---------- ----------
abstract.high_fanout_limit integer  --        --                 600          block     normal
. . .
file.verilog.default_user_label    string    --       {}        --          global    normal
file.verilog.keep_unconnected_cells bool      --                 true         global    normal
------------------------------------ ---------- ---------- ------------ -------------- ---------- ---------- ----------
1
```

The following example reports application options, that are global scoped and have values different from the default.

```
prompt> report_app_options -global -non_default
****************************************
Report : app_option
 ...
****************************************

Name                Type      Value     User-default System-default Scope     Status    Source
---------------------- ---------- ---------- ------------ -------------- ---------- ---------- --------------------
test.global_only_bool  bool      true      true        false         global    normal
test.global_only_int   integer   12        12          10            global    normal    app_option.tcl:1151
---------------------- ---------- ---------- ------------ -------------- ---------- ---------- --------------------
1
```

The following example reports the application options using the '-as_list' option.

```
prompt> report_app_options -as_list
****************************************
Report : app_option
 ...

****************************************
abstract.high_fanout_limit 600
\...
test.layer_name_number_list_new {M1 31}
verilog.default_user_label {}
1
```

The following example illustrates reporting using the '-as_list' option when the user and system defined value is absent.

```
prompt> reset_app_options -user_default opt.power.mode
1
prompt> reset_app_options opt.power.mode
1
prompt> report_app_options opt.power.mode
****************************************
Report : app_option
Design : r4000
Version: M-2016.12-DEV
Date   : Wed Feb 24 05:43:30 2016
****************************************
Name          Type   Value    User-default    System-default    Scope
---------------- ------ --------- ---------------- ------------------ --------
opt.power.mode   enum   {}       {}              none              block
```

```
---------------- ------ --------- ---------------- ------------------ --------
1
prompt> report_app_options opt.power.mode -as_list
****************************************
Report : app_option
Design : r4000
Version: M-2016.12-DEV
Date   : Wed Feb 24 05:43:37 2016
****************************************
No options matched the specified pattern(s)
1
```

The following example reports the application options using the '-csv' option.

```
prompt> report_app_options -csv report.csv

report.csv
```

SEE ALSO

get_app_option_value(2)
get_app_options(2)
set_app_options(2)
reset_app_options(2)
help_app_options(2)

report_app_var

Shows the application variables.

SYNTAX

string **report_app_var**
[-verbose]
[-only_changed_vars]
[*pattern*]

Data Types

pattern string

ARGUMENTS

-verbose

Shows detailed information.

-only_changed_vars

Reports only changed variables.

pattern

Reports on variables matching the pattern. The default is "*".

DESCRIPTION

The **report_app_var** command prints information about application variables matching the supplied pattern. By default, all descriptive information for the variable is printed, except for the help text.

If no variables match the pattern, an error is returned. Otherwise, this command returns the empty string.

If the **-verbose** option is used, then the command also prints the help text for the variable. This text is printed after the variable name and all lines of the help text are prefixed with " #".

The Constraints column can take the following forms:

{val1 ...}

The valid values must belong to the displayed list.

val \<= a

The value must be less than or equal to "a".

val \>= b

The value must be greater than or equal to "b".

b \<= val \<= a

The value must be greater than or equal to "b", and less than or equal to "a".

EXAMPLES

The following are examples of the **report_app_var** command:

```
prompt> report_app_var sh*
Variable              Value    Type    Default   Constraints
------------------------ --------- ------- --------- ----------------------
sh_continue_on_error    false    bool    false
sh_script_stop_severity  none     string  none     {none W E}
 \.\.\.


prompt> report_app_var sh* -verbose
Variable              Value    Type    Default    Constraints
------------------------- --------- ------- --------- ----------------------
sh_continue_on_error    false    bool    false
  # Allows source to continue after an error
sh_script_stop_severity  none     string  none     {none W E}
  # Indicates the error message severity level which would cause
  # a script to stop executing before it completes
 \.\.\.
```

SEE ALSO

get_app_var(2)
set_app_var(2)
write_app_var(2)

report_area

Displays area information for the current design or instance.

SYNTAX

status **report_area**
[-nosplit]
[-significant_digits digits]
[-physical]
[-hierarchy]
[-designware]
[-logical]

ARGUMENTS

-nosplit

Prevents line splitting. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are reported. Allowed values are from 0 through 13. The default value is 2. Using this option overrides the value set by the **shell.common.report_default_significant_digits** app option.

-physical

Reports the size of the core area and the aspect ratio of the design.

-hierarchy

Reports the area used by cells across the design hierarchy. Reports the absolute value and the percentage of area consumed by each of the cells across the hierarchy. This option also reports the details of area contribution by combinational, non-combinational, and macro or black box cells. The "black boxes" column includes macro area.

-designware

Reports the area of synthetic cells. There are two types of synthetic cells:

- Datapath cells
  The datapath cells are extracted complex datapath cells.
- DesignWare singleton cells
  The DesignWare singleton cells are instantiated or inferred synthetic component cells.

The report shows the total synthetic cell area and the total datapath cell area. If the synthetic cells are ungrouped during compile, the report shows the estimated area of the ungrouped synthetic cells.

-logical

Reports the area using logical area. If there are lib cells without logical area, physical area will be used instead. And a warning message will be printed out.

DESCRIPTION

The **report_area** command lists the area statistics for the current instance or the current design. The report includes combinational, non-combinational, and total area information. If you set the **current_instance** command, the report is generated for the design of that instance. Otherwise the report is generated for the current design.

The number of combinational cells, number of sequential cells, and the number of macros/black boxes only include leaf cells. For the purposes of the **report_area** command, macros and leaf-level black box cells are reported together. Hierarchical black-box cells are not counted because no area is associated with them.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example generates an area report:

```
prompt> report_area

****************************************
Report : area
 ...

****************************************

Number of cells:                  256
Number of combinational cells:        254
Number of sequential cells:           0
Number of macros/black boxes:         0
Number of buf/inv:                   31
Number of references:                23

Combinational area:          228294.4006
Buv/Inv area:                21384.0142
Noncombinational area:           0.0000
Macro/Black Box area:            0.0000

Total cell area:             228294.4006
1
```

The following example generates an area report for a hierarchical design using the **-hierarchy** option:

```
prompt> report_area -hierarchy
```

```
****************************************
Report : area
Design : top
Version: 2.1.0
Date   : Tue May 17 14:07:30 2016
****************************************

Number of cells:                  256
Number of combinational cells:        254
Number of sequential cells:           0
Number of macros/black boxes:         0
Number of buf/inv:                   31
Number of references:                23

Combinational area:          228294.4006
Buv/Inv area:                21384.0142
Noncombinational area:           0.0000
Macro/Black Box area:            0.0000

Total cell area:             228294.4006

Hierarchical area distribution
------------------------------

                  Global cell area          Local cell area
                  ------------------- -------------------------------
Hierarchical cell      Absolute    Percent Combi-      Noncombi- Black-
                       Total       Total   national    national  boxes   Design
----------------------- ----------- ------- ----------- --------- ------ --------------------------------
top                    228294.5469   100.0  12633.5693    0.0000  0.0000 top
U1                     123539.4844    54.1   3967.9995    0.0000  0.0000 test_sel_1
U1/add_x_16_1            4550.4004     2.0   4550.4004    0.0000  0.0000 test_sel_1_DW01_add_30
U1/add_x_18_1            4998.3999     2.2   4998.3999    0.0000  0.0000 test_sel_1_DW01_add_29
U1/add_x_25_1            6054.4004     2.7   6054.4004    0.0000  0.0000 test_sel_1_DW01_add_22
U1/add_x_25_2            4390.4004     1.9   4390.4004    0.0000  0.0000 test_sel_1
U1/add_x_16_1            4550.4004     2.0   4550.4004    0.0000  0.0000 test_sel_1_DW01_add_30
U1/add_x_18_1            4998.3999     2.2   4998.3999    0.0000  0.0000 test_sel_1_DW01_add_29
U1/add_x_25_1            6054.4004     2.7   6054.4004    0.0000  0.0000 test_sel_1_DW01_add_22
U1/add_x_25_2            4390.4004     1.9   4390.4004    0.0000  0.0000 test_sel_1_DW01_add_12
U1/mult_x_10_1          21939.1992     9.6  21939.1992     0.0000  0.0000 test_sel_1_DW02_mult_J1_3
U1/mult_x_11_1          22195.1973     9.7  22195.1973     0.0000  0.0000 test_sel_1_DW02_mult_J1_2
U1/mult_x_12_1          21913.5898     9.6  21913.5898     0.0000  0.0000 test_sel_1_DW02_mult_J1_1
U1/mult_x_13_1          22732.8008    10.0  22732.8008     0.0000  0.0000 test_sel_1_DW02_mult_J1_0
U1/sub_x_17_1            5216.0000     2.3   5216.0000    0.0000  0.0000 test_sel_1_DW01_sub_6
U1/sub_x_19_1            5580.8008     2.4   5580.8008    0.0000  0.0000 test_sel_1_DW01_sub_7
U2                      92121.9531    40.4   3276.7986    0.0000  0.0000 test_sel_0
U2/DP_OP_15J1_64_71     40691.3789    17.8  40691.3789    0.0000  0.0000 test_sel_0_DP_OP_15J1_64_71_0
U2/DP_OP_16J1_65_71     39392.1328    17.3  39392.1328    0.0000  0.0000 test_sel_0_DP_OP_16J1_65_71_0
U2/DP_OP_17J1_66_1425    8761.5996     3.8   8761.5996    0.0000  0.0000 test_sel_0_DP_OP_17J1_66_1425_1
----------------------- ----------- ------- ----------- --------- ------ --------------------------------
Total                                        228294.6562    0.0000  0.0000
1
```

The following example uses the **-designware** option to generate an area report for a design that contains synthetic cells:

```
prompt> report_area -designware
```

```
****************************************
Report : area
Design : top
Version: 2.2.0
Date   : Thu Nov  3 07:57:41 2016
****************************************

Number of cells:                  125
Number of combinational cells:          125
Number of sequential cells:               0
Number of macros/black boxes:             0
Number of buf/inv:                   19
Number of references:                26

Combinational area:              1052.39
Buf/Inv area:                    69.59
Noncombinational area:               0.00
Macro/Black Box area:                0.00

Total cell area:                 1052.39

Area of detected synthetic parts
--------------------------------
 No DW parts to report!

Estimated area of ungrouped synthetic parts
-------------------------------------------
                              Estimated   Perc. of
Module              Implem.  Count      Area  cell area
-----------------------------------------------------------------
DP_OP_11_3838_33278_J1  str        1     130.70     12.4%
DW01_cmp2           pparch       2     181.01     17.2%
DW_mult_uns         pparch       1     551.65     52.4%
-----------------------------------------------------------------
Datapath Subtotal:                 1     130.70     12.4%
Total:                         4     863.36     82.0%

Subtotal of datapath cell area: 130.70  12.4%  (estimated)
Total synthetic cell area:      863.36  82.0%  (estimated)
1
```

SEE ALSO

report_qor(2)
report_design(2)

report_attachments

Reports the details of the given attachments from the specified owner object.

SYNTAX

string **report_attachments**
names
[-of_object *object*]

Data Types

names list
object string or collection

ARGUMENTS

names

List of attachment names to be reported. Supports glob style patterns.

-of_object *objects*

Specifies the object whose attachments will be reported. By default current block and current library will be considered in that order as owner object.

DESCRIPTION

This command reports the attachments with the names in the given list from the objects provided through the *-of_object* option. If *-of_object* is not used, then the current block is used as the owner object. If current block is not set, current library is used as the owner object For the attachment names, Glob style pattern is supported.

EXAMPLES

The following example removes the attachment test_att from the current block.

```
prompt> report_attachments -of_object [current_block] test_attachment
############################################################
#
```

```
# Attachments Report
# Owner Design : ORCA_TOP
#
################################################################

 Multi-Section Name
 ------------ ------------------------------------------------
    No      test_attachment

################################################################
test_attachment
```

SEE ALSO

add_attachment(2)
remove_attachments(2)
open_attachment(2)

report_attribute

Reports the attributes on one or more objects.

SYNTAX

string **report_attributes**
[-class *class_name*]
[-nosplit]
[-application]
[-compact]
object_spec

Data Types

class_name string
object_spec list

ARGUMENTS

-class *class_name*

If object_spec is a name, this is its class. Allowable values are *design*, *port*, *cell*, *pin*, *net*, *lib*, *lib_cell*, or *lib_pin*.

-nosplit

Does not split lines if column overflows.

-application

Lists application attributes as well as user-defined attributes.

-compact

Shortens design/object name strings to fit within defined column width.

object_spec

List of objects to report. Each element in the list is either a collection or a pattern which combines with the *class_name* to find the objects.

DESCRIPTION

Generates a report of attributes on the specified objects. By default, only user-defined attributes display. Using the **-application** option adds application attributes to the report. For application attributes which are of the type *collection*, the name first of the first object in the collection will be displayed.

Some application lib_cell, lib_pin, terminal, shape, and layer attributes may be overridden by the user. These overrides are scoped to and stored persistently in the current design. This command prints the string "(override)" next to the attribute value if the value is one of these design-specific overrides.

EXAMPLES

This example defines an attribute 'X' for cells, then sets the value on all cells in this level of the hierarchy, and reports the result.

```
prompt> define_user_attribute -type int -class cell X
prompt> set_user_attribute [get_cells *] X 30
Set attribute 'X' on 'i1'
Set attribute 'X' on 'i2'

prompt> report_attributes [get_cells *]
****************************************
Report : Attribute
Design : my_des
****************************************

Design    Object      Type    Attribute Name   Value
-------------------------------------------------------------
my_des    i1          int     X                30
my_des    i2          int     X                30
```

This example displays application attributes.

```
prompt> report_attributes -application [get_designs my_des]
****************************************
Report : Attribute
Design : my_des
****************************************

Design  Object   Type     Attribute Name    Value
-----------------------------------------------------
my_des  my_des   string   full_name         my_des
my_des  my_des   string   object_class      design
my_des  my_des   float    area              56.000000
my_des  my_des   string   tree_type_max     balanced_case
my_des  my_des   string   tree_type_min     balanced_case
my_des  my_des   float    wire_load_min_block_size
                          0.000000
my_des  my_des   string   wire_load_mode    top
```

This example displays the report with compacted design and object names.

```
prompt> report_attributes -app -compact [get_vias my_long_via_name]
****************************************
Report : Attribute
Design : my_long_design_name
****************************************
```

```
Design        Object        Type      Attribute Name ...
-------------------------------------------------------
my_lon...n_name my_lon...a_name string    array_size    ...
my_lon...n_name my_lon...a_name rect      bbox          ...
my_lon...n_name my_lon...a_name collection bounding_box  ...
```

SEE ALSO

collections(2)
define_user_attribute(2)
get_attribute(2)
list_attributes(2)
remove_attributes(2)
define_user_attribute(2)

report_attributes

Reports the attributes on one or more objects.

SYNTAX

string **report_attributes**
[-class *class_name*]
[-nosplit]
[-application]
[-compact]
object_spec

Data Types

class_name string
object_spec list

ARGUMENTS

-class *class_name*

If object_spec is a name, this is its class. Allowable values are *design*, *port*, *cell*, *pin*, *net*, *lib*, *lib_cell*, or *lib_pin*.

-nosplit

Does not split lines if column overflows.

-application

Lists application attributes as well as user-defined attributes.

-compact

Shortens design/object name strings to fit within defined column width.

object_spec

List of objects to report. Each element in the list is either a collection or a pattern which combines with the *class_name* to find the objects.

DESCRIPTION

Generates a report of attributes on the specified objects. By default, only user-defined attributes display. Using the **-application** option adds application attributes to the report. For application attributes which are of the type *collection*, the name first of the first object in the collection will be displayed.

Some application lib_cell, lib_pin, terminal, shape, and layer attributes may be overridden by the user. These overrides are scoped to and stored persistently in the current design. This command prints the string "(override)" next to the attribute value if the value is one of these design-specific overrides.

EXAMPLES

This example defines an attribute 'X' for cells, then sets the value on all cells in this level of the hierarchy, and reports the result.

```
prompt> define_user_attribute -type int -class cell X
prompt> set_user_attribute [get_cells *] X 30
Set attribute 'X' on 'i1'
Set attribute 'X' on 'i2'

prompt> report_attributes [get_cells *]
****************************************
Report : Attribute
Design : my_des
****************************************

Design    Object      Type    Attribute Name   Value
-------------------------------------------------------------
my_des    i1          int     X                30
my_des    i2          int     X                30
```

This example displays application attributes.

```
prompt> report_attributes -application [get_designs my_des]
****************************************
Report : Attribute
Design : my_des
****************************************

Design  Object   Type     Attribute Name    Value
-----------------------------------------------------
my_des  my_des   string   full_name         my_des
my_des  my_des   string   object_class      design
my_des  my_des   float    area              56.000000
my_des  my_des   string   tree_type_max     balanced_case
my_des  my_des   string   tree_type_min     balanced_case
my_des  my_des   float    wire_load_min_block_size
                          0.000000
my_des  my_des   string   wire_load_mode    top
```

This example displays the report with compacted design and object names.

```
prompt> report_attributes -app -compact [get_vias my_long_via_name]
****************************************
Report : Attribute
Design : my_long_design_name
****************************************
```

```
Design        Object        Type      Attribute Name ...
-------------------------------------------------------
my_lon...n_name my_lon...a_name string    array_size    ...
my_lon...n_name my_lon...a_name rect      bbox          ...
my_lon...n_name my_lon...a_name collection bounding_box  ...
```

SEE ALSO

collections(2)
define_user_attribute(2)
get_attribute(2)
list_attributes(2)
remove_attributes(2)
define_user_attribute(2)

report_auto_floorplan_constraints

Sets constraints for initializing floorplan implicitly.

SYNTAX

status **report_auto_floorplan_constraints**
[-control_type]
[-core_utilization]
[-shape]
[-orientation]
[-side_ratio]
[-side_length]
[-core_offset]
[-row_core_ratio]
[-coincident_boundary]
[-boundary]
[-flip_first_row]
[-pin_snap]
[-honor_pad_limit]
[-site_def]
[-all]
[-use_site_row]

Data Types

ARGUMENTS

-control_type

Reports the control type of the core or die. The default is ratio.

-core_utilization

Reports the core utilization. The default is 0.7.

-shape

Reports the core boundary shape. The default is rectangular (R).

-orientation

Reports the orientation of the specified core shape.

-side_ratio

Reports the ratio of the edges of the core shape.

-side_length

Reports the length of the edges of the core shape.

-core_offset

Reports the distance between the side of the core and boundary.

-row_core_ratio

Reports the channel ratio between cell rows. The default is 1.0.

-coincident_boundary

Reports whether to keep the same existing boundary.

-boundary

Reports the boundary of the core shape.

-flip_first_row

Reports whether the command flips the first row at the bottom of the core area for horizontally placed cell rows or flips the leftmost row for vertically placed cell rows. By default, this option is enabled.

-pin_snap

Reports whether to snap terminal centers to the center of wire tracks. By default, this setting is disabled.

-honor_pad_limit

Reports whether to adjust the core and die size to honor pad-limited designs.

-site_def

Reports the site def name used.

-all

Reports all the constraint settings used by auto floorplanning.

-use_site_row

Reports whether to create site rows.

DESCRIPTION

This command reports the constraints that create a floorplan with a boundary, core, site arrays (or rows), and wire tracks. Before executing this command, you must open a physical design by using the **open_block** command or create a design by using the **read_verilog** or **read_verilog_outline** command.

EXAMPLES

The following example reports the constraint of utilization to be 0.8.

```
prompt> report_auto_floorplan_constraints -core_utilization
1
```

The following example reports the preferred core shape to be rect (R).

```
prompt> report_auto_floorplan_constraints -shape
1
```

The following example reports the preferred core length to create the floorplan.

```
prompt> report_auto_floorplan_constraints -side_length
1
```

SEE ALSO

report_auto_floorplan_constraints(2)
create_io_ring(2)
remove_io_rings(2)
report_io_rings(2)

report_auto_save

Reports the auto-saved information of a library.

SYNTAX

status **report_auto_save**
[-library library_name | library_path]

Data Types

library_path string
library_name string

ARGUMENTS

-library *library_name* | *library_path*

Specifies the library name or path whose auto-saved information needs to be reported.

DESCRIPTION

This command reports auto-saved information auto-saved library name, version, disk-size, location, the time-stamp of each auto-saved library of the specified library.

By default, the command reports auto-saved information of the current library.

EXAMPLES

```
prompt> report_auto_save
************************************************************
Original Library : r4000

Auto-saved library path              : /slowfs/ndmdisk003/r4000_19142
Auto-saved library timestamp         : Wed Aug 14 00:54:44 2019
Auto-saved library size              : 2912391 bytes
Auto-saved library is_recovery_save  : false
Auto-saved library context           : create_net
```

SEE ALSO

stop_auto_save(2)
recover_auto_save(2)
remove_auto_save(2)
start_auto_save(2)

report_background_jobs

Reports all the completed and running background jobs submitted to run by the **redirect -bg** command.

SYNTAX

status **report_background_jobs**
[-reset]

ARGUMENTS

-reset

Removes the completed jobs from the list of tracked jobs.

DESCRIPTION

This command reports all the jobs that were submitted to run by the **redirect** command with the **-bg** option.

All the completed and running background job are reported.

To omit the completed jobs, use the **-reset** option with the **report_background_jobs** command.

EXAMPLES

The following example shows a **report_background_jobs** run.

```
prompt> redirect -bg -max_cores 1 -file rep.out {source rep.tcl}
prompt> report_background_jobs
```

SEE ALSO

redirect(2)

report_block_pin_constraints

Reports block pin constraints for blocks in the design.

SYNTAX

string **report_block_pin_constraints**
[-cells *cell_collection*]
[-self]
[-pin_spacing_control]

Data Types

cell_collection collection

ARGUMENTS

-cells *cell_collection*

Restricts the block pin constraints report to only the specified collection of cells. By default, the command reports pin constraints on all blocks.

-self

Restricts the block pin constraints report to only the top-level design. By default, the command reports pin constraints for all blocks in the design.

-pin_spacing_control

If this option is specified, the command report the block pin spacing constraints that are set by read_pin_constraints and associated with reference blocks.

DESCRIPTION

This command reports the block pin constraints on every level. If you do not specify any options, the command reports the block pin constraints for all blocks.

EXAMPLES

The following example displays the block pin constraints on cell I_CLOCK_GEN.

```
prompt> report_block_pin_constraints -cells [get_cell I_CLOCK_GEN]
```

SEE ALSO

remove_block_pin_constraints(2)
set_block_pin_constraints(2)

report_block_shaping

Generates a quality of results (QoR) report for block shaping.

SYNTAX

status **report_block_shaping**
[-utilization_from_target]
[-core_area_violations]
[-overlaps]
[-chimney_area]
[-detour_estimate]
[-unaligned_pins_estimate]
[-flyline_crossing]
[-orientation_violations]
[-hierarchical]
[-channel *height*]
[-error_view *view_name*]
[-verbose minimum | low | high]
[-cells *collection_of_blocks*]

Data Types

height distance
view_name string
collection_of_blocks collection of block instances

ARGUMENTS

-utilization_from_target

Reports the difference in utilization between the actual utilization of the block and its target utilization as determined by the target_utilization attribute.

-core_area_violations

Reports blocks that are placed outside the core area.

-overlaps

Reports blocks that overlap.

-chimney_area

Reports the area used by chimneys. Chimneys are long, narrow channels which protrude from blocks (like a bump). Chimneys can create less than optimal placement.

-detour_estimate

Reports the estimated number of net detours caused by blocks, assuming that the blocks do not allow feedthroughs. A net detour is a net that is routed outside of its bounding box. Use this option after top-level placement. If top-level cells are not placed, this report is inaccurate.

-unaligned_pins_estimate

Reports the estimated number of unaligned pins. This estimate is based on an ideal pin assignment result; the estimate is a lower bound on actual number of unaligned pins after pin assignment.

-flyline_crossing

Reports the number of block-to-block flyline crossings. For example, suppose there is a block pair, (block A, block B) and there are 5 flylines between block A and B, and suppose there is another block pair, (block C, block D) and there are 4 flylines between block C and D, then if these flylines cross, the number of block-to-block flyline crossings would be 5x4=20. This report assumes that block flylines originate from the center of each block.

-orientation_violations

Report the orientation violations of blocks in the design. The current orientation of a block is considered as its orientation relative to the orientation of its parent block. When the current orientation of the block is not one of its legal orientations, the report lists the name of the block, its current orientation and legal orientations. The legal orientations can be set by using the **set_attribute -name allowable_orientation** command. The report also shows the number of cells with orientation violation in each design and the total number of violations.

-hierarchical

Reports block shaping information for the current block and for subblocks. By default, the command reports block shaping information only for the current block.

-channel *height*

Reports the area of channels whose height is less than *height*. Only blocks and core area are considered. This is useful to check if when manually shaping an abutted design, no channel was accidentally created.

-error_view *view_name*

Specifies the name of the generated error view. The command saves the error view data to a file in XML format with the specified name. No attachment is created nor saved in to the design database. If the view_name file already exists under the current run directory, the command overwrites the file. If a file extension is specified, it must be ".err", otherwise it is ignored. If no extension is specified, the ".err" extension is appended to the file name.

-verbose minimum | low | high

Controls the number of messages generated by the report. The *minimum* argument generates the least amount of information; *low* generates a one to two page report; *high* generates the longest report. By default, the verbosity level is low.

-cells *collection_of_blocks*

Specifies the blocks to analyze. By default, all blocks are analyzed.

DESCRIPTION

This command reports of the quality of the block shaping result. Most of the options can be used after block shaping, however, the **-detour_estimate** option requires top-level cells to be placed to give an accurate result.

EXAMPLES

The following example generates a QoR report for the current block shaping result.

```
prompt> report_block_shaping -overlaps

****************************************
Report : report_block_shaping
 ...

****************************************

 Block overlap report
 ====================
 Block overlaps in design orca: 0
```

SEE ALSO

create_placement(2)
gui_remove_all_annotations(2)
shape_blocks(2)

report_block_to_top_map

Reports the relationship between top-level mode, corner, and clock objects to other objects at lower levels of hierarchy.

SYNTAX

int **report_block_to_top_map**
[-blocks *cells*]
[-path *hierarchy_name*]
[-waveform]
[-warnings]

Data Types

cells string or collection
hierarchy_name string

ARGUMENTS

-blocks *cells*

Reports only on mapped modes, corners, and clocks in the specified blocks. By default, all mappings are reported.

-path *hierarchy_name*

Reports only on mapped modes, corners, and clocks in the block with the specified path. It is not necessary for the block to be loaded into memory. By default, all mappings are reported.

-waveform

Displays the waveform for the clock together with each block-level clock. By default, waveforms are not displayed. To show the waveform of top-level clocks, use the **report_clocks** command.

-warnings

Reports common problems that can occur in clock mapping. These include:

- Block-level clocks that are not mapped to a top-level clock and are not marked as an unused_clock.
- Block-level clocks that are not electrically connected to their associated top-level clock.
- Block-level clocks that have a different waveform than their associated top-level clock.
- Block-level clocks that are driven by top-level clocks that are declared on the block boundary (boundary clocks). These clocks might have been created with **set_block_to_top_map -auto_clock all**. The top-level boundary clock allows timing analysis to be done, but it is not necessarily correct for signoff. You should consider removing or replacing this clock before signoff.

DESCRIPTION

Reports the relationship between top-level mode, corner, and clock objects to objects inside of instances of lower level designs. These associations are used by commands such as **write_io_constraints**. The associations are set using the **set_block_to_top_map** command. Write out block-to-top mappings by using the **write_script** command, so they can be easily reapplied to new netlists.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following example reports block-to-top mappings for all modes, corners, and blocks.

```
prompt> report_block_to_top_map
```

The following example reports the block-to-top mappings only for block "CPU". corners, and blocks.

```
prompt> report_block_to_top_map -blocks CPU
```

The following example reports warnings about your mapped clocks.

```
prompt> report_block_to_top_map -warnings
```

SEE ALSO

report_clocks(2)
report_corners(2)
report_modes(2)
set_block_to_top_map(2)
write_io_constraints(2)
write_script(2)

report_boundary_cell_rules

Reports the boundary cell placement rules.

SYNTAX

status **report_boundary_cell_rules**

ARGUMENTS

None.

DESCRIPTION

This command is used to report the boundary cell placement rules set by *set_boundary_cell_rules*. For those rules user want to use default value and doesn't specify in *set_boundary_cell_rules*, this command will not report them.

EXAMPLES

The following example reports boundary cell placement rules for left and right boundary cells.

```
prompt> set_boundary_cell_rules -left_boundary_cell myLib/CellLeft \
        -right_boundary_cell myLib/CellRight
prompt> report_boundary_cell_rules
```

SEE ALSO

set_boundary_cell_rules(2)
remove_boundary_cell_rules(2)
compile_boundary_cells(2)
check_boundary_cells(2)
compile_targeted_boundary_cells(2)

check_targeted_boundary_cells(2)

report_boundary_optimization

Reports the boundary_optimization restrictions on pins, cells or modules only for object specific controls and not during global control "compile -no_boundary_optimization". This command is valid for “set_boundary_optimization” and not for “compile -no_boundary_optimization”

SYNTAX

int **report_boundary_optimization**
[object_list]

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of pins, cells or modules whose boundary optimization restrictions to be reported. Pin, cell or module names in object_list must be from the current design. If more than one object is specified, they must be enclosed in quotation marks ("") or braces ({}).

DESCRIPTION

This command reports the boundary_optimization restriction on objects in object_list. If the object_list is not specified, the command reports boundary optimization restrictions on the objects in the current design. The command reports the information on the objects which are restricted by set_boundary_optimization command.

EXAMPLES

The following example shows how to report previously set boundary optimization restriction on the cells named U0, U1 and on the pin named U2/A.

prompt> report_boundary_optimization { U0 U1 U2/A}

The following example shows how to report all previously set boundary optimization restrictions in the current design.

prompt> report_boundary_optimization

SEE ALSO

compile(2)
set_boundary_optimization(2)
remove_boundary_optimization(2)

report_bounds

Reports bounds in the current design.

SYNTAX

int **report_bounds**
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*bound_list*]

Data Types

digits int
bound_list list

ARGUMENTS

-verbose

Displays verbose bound information.

-nosplit

Does not split lines if column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default. Fixed-precision floating-point arithmetics is used for delay calculation; therefore, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h. The upper bound on a meaningful value of the argument to **-significant_digits** is then FLT_DIG - ceil(log10(fabs(*any_reported_value*))).

bound_list

Specifies a list of bounds to report. The list may contain bound names, patterns, or collections. A collection may be specified by using the **get_bounds** command.

DESCRIPTION

The **report_bounds** command generates a report about the bound constraints in the design. The report includes the bound name and a description of the bound. If **-verbose** is specified, the report also includes bound region coordinates, dimensions, and cells and ports assigned to the bound.

If *bound_list* is specified, those bounds will be reported. If it is not specified, then all bounds in the design will be reported.

EXAMPLES

The following example reports all bounds in the design.

```
prompt> report_bounds *
Bound           Description
------------------------------------------------------------------------
groupBound1     medium effort auto group bound
groupBound2     soft group bound
groupBound3     ultra effort auto group bound
moveBound1      soft move bound
moveBound2      hard move bound
moveBound3      hard exclusive move bound
1
```

The following example reports verbose information for the bound named "groupBound2".

```
prompt> report_bounds -verbose groupBound2*
Bound           Description
------------------------------------------------------------------------
groupBound2     soft group bound
                Dimensions: (1000.00, 2000.00)
                Cells:    inv1
                          inv2
                          inv3
                Ports:    reset
1
```

SEE ALSO

create_bound(2)
get_bounds(2)
add_to_bound(2)
remove_from_bound(2)
remove_bounds(2)

report_budget

Reports timing budget information for the current design.

SYNTAX

int **report_budget**
[-blocks]
[-html *file_name*]
[-html_dir *directory_name*]
[-latency]
[-pins *budget_pin_list*]
[-fanin_segments *fanin_pin*]
[-fanout_segments *fanout_pin*]
[-through *through_pin*]
[-hold_through *through_pin*]
[-busplans *busplans*]
[-input]
[-output]
[-warning_pins]

Data Types

file_name string
directory_name string
budget_pin_list collection of pins
fanin_pin collection of pins
fanout_pin collection of pins
through_pin collection of pins
busplans collection of busplans

ARGUMENTS

-blocks

List the blocks that were selected for budget processing by the **set_budget_options** command. Also, show specific options that were selected for each block.

-html *file_name*

Writes a budget report file in HTML format. You can view the HTML file in an HTML browser such as Internet Explorer, Firefox, or Chrome. The file contains almost every detail about your budget and your circuit's block boundary timing in the current mode. The report includes hyperlinks to help you navigate between the various report sections. For example, the report for a budget pin is linked to the report for its connected budget segments. See the DESCRIPTION section for details about the contents of the report sections. The report sections included in the HTML report are: block summaries, segment lists, clock summaries, path summaries,

pin summaries, and segment summaries.

Specify .html as the file extension for *file_name*, as this is required by most browsers. Note that files produced by *-html* can be very large and most browsers do no perform well on large files. If you encounter problems due to a large report file, try using the *-html_dir* option.

-html_dir *directory_name*

Writes multiple budget report files in HTML format to the specified directory. The output is split across many files in the specified directory to allow easier browsing and to avoid overwhelming the browser with one large file. The command creates the index.html file in the specified directory to contain a table of contents that references the individual files.

The *-html_dir* option outputs budget data for all active modes; the *-html* option processes only the current mode.

-latency

Show a summary of the budgeted clocks in your design and where they cross block boundaries. Each time a clock crosses a block boundary, a new line is generated in the report. The line in the report shows the median clock latency and the fanout for that particular part of the clock tree. This raw information is used to calculate the relative clock latency in each block -- which is needed during budgeting.

The latency report lists clocks in "clock groups". Clocks that have datapaths crossing between them are put in the same group. Clocks that are completely independent are placed in separate groups.

-pins *budget_pin_list*

Prints a pin summary for the given budgeted pins. If the specified pin is not on a budget block, no information is reported. The contents of a pin summary are described in the DESCRIPTION section.

-fanin_segments *fanin_pin*

Prints a segment summary for the budget segment objects that fan into the specified *fanin_pin*. If the specified *fanin_pin* is not on a budget block, no information is reported. The contents of a segment summary are described in the DESCRIPTION section.

-fanout_segments *fanout_pin*

Prints a segment summary for the budget segment objects that fan out from the specified *budget_pin*. If the specified *fanout_pin* pin is not on a budget block, no information is reported. The contents of a segment summary are described in the DESCRIPTION section.

-through *through_pin*

Reports the budget of the worst-slack path through the specified budget pins. If the specified pin is not on a budget block, no information is reported. A worst path is shown for each unique "from clock" and "to clock". There might be other, less critical paths through the pin that are not shown. If you want to look at the less critical paths, you must trace the paths manually. The output of the *-html* option is useful for tracing arbitrary budget paths.

The format of the report output is similar to that of the pin summary and segment summary sections. The contents of a pin summary are described in the DESCRIPTION section.

-hold_through *through_pin*

Reports the hold budget of the worst hold slack path (the fastest path) through the specified budget pin. If the specified pin is not on a budget block, no information is reported. A worst path is shown for each unique "from clock" and "to clock". There might be other, less critical paths through the pin that are not shown. If you want to look at the less critical paths, you must trace the paths manually. The output of the *-html* option is useful for tracing arbitrary budget paths.

The format of the report output is similar to that of the pin summary and segment summary sections. The contents of a pin summary are described in the DESCRIPTION section.

-busplans *busplans*

For each of the given busplans, show the segments in the budget which are covered. The delay shown is the delay that will be applied if you use the *-busplans* option of the **compute_budget_constraints** command. Please refer to the **create_budget_busplan** man page for more details.

-input

Limits the report to information only about input budget pins. This option is also useful when your circuit has "inout" pins and you want to look at the input direction.

-output

Limits the report to information only about output budget pins. This option is also useful when your circuit has "inout" pins and you want to look at the output direction.

-warning_pins

Reports additional detail about potential problems in the design or budget. This option adds information to the report regarding "Budgeted pins with seemingly impossible constraints", "Pins with missing budget constraints", and "Pins with no timing paths".

DESCRIPTION

This command outputs reports that show detail about your budget and your current circuit's block boundary timing. Almost all of the information reported by the **report_budget** command is available by using the **get_budgets** command and **budget_attribute** attributes. Use the **report_budget** command to produce a well-formatted report that contains budget information.

The **report_budget** command reports different information based on the options you specify. Some options show you multiple categories of information as described in the following sections.

Block Report

Budget segments going into and coming out of a budget block are categorized and summarized. The segment categories include groups such as "To input pin of block from top level startpoint", or "From output pin of block to input pin of block B2", or "To input pin of block from output pin of block B3". For each segment category, information includes "worst negative budget slack" (WNS), "total negative budget slack" (TNS), "count" of the budget segments, and "number of segments that violate the budget" (NVIO).

Segment List

For each segment category in the block report, a list shows detail about the member segments of the category. The top 50 worst-slack segments are shown. Each segment has one line of information from the full segment summary report. In the HTML report, a hyperlink directs you to the segment summary.

Clock Summary

A clock summary shows the following information:

- Data for each time one of your system clocks crosses a block boundary
- The current budget settings from the **set_latency_budget_constraints** command
- The current clock latency and fanout downstream from the block crossing
- The latency used by budget process when it calculates datapath lengths

Path Summary

For each set of datapaths that cross from one clock across a budget boundary to another clock, a summary is printed. The report

shows the launch and capture clock, along with any multicycle path exception that might exist. In the HTML report, a hyperlink links to the launch and capture clock summaries.

The path summary also includes details about the budget calculations for data that has this path description, including calculations for: the path delay along paths of this type, the clock_uncertainty applied to paths launching in a block and going to a block output, the clock_uncertainty applied to paths starting at a block input and ending in a block.

Pin Summary

For each budget pin in the design, a summary tells you about the budgets that have been applied to the pin. There is a separate summary for each type of datapath that crosses at the pin. The HTML report contains a link to the "path summary". The pin summary shows the user constraint that has been applied by using the **set_pin_budget_constraints** command or from the **compute_budget_constraints** command.

The pin summary also includes three lines that describe the timing of the "full budget path", the timing of the "fanin budget" (the part of the path before the pin), and the timing of the "fanout budget" (the part of the path that comes after the pin). Each line has the following format:

(S= 1.545 / D= 6.255(2.16) / B= 7.800) 36%

The "D" (6.255) number is the actual timed delay of the path or portion of the path, in nanoseconds. The number in parentheses (2.16) is the portion of the path delay that the budgeter treats as "fixed" delay. The "B" number (7.800) is the amount of time that the budgeter has allocated for the path or portion of the path. If no budget has been set, the report shows "*none*", instead of a number. The S number (1.545) is the budget slack. The budget slack is simply calculated as "B minus D". If a percentage number (36%) is shown, the number indicates the fraction of the whole-path budget that has been assigned to this portion of the path.

The pin summary also shows one line for each budget segment that fans into or out of the pin. In the HTML report, there is a hyperlink to the segment summary.

For hold budgeting, the pin summary simply shows the "hold arrival" time that the budgeter will require in the constraints. For example, the report might show:

Budgeted hold arrival (rise/fall):
0.010/0.010 (Full path: 0.020 corner: min)

This means that the full path must be at least 0.020 ns long in order to meet the top-level hold time constraint. And a budget will be created such that the arrival time at the reported pin must be at least 0.010 ns. You should use the "plan.budget.write_hold_budgets" app option to activate writing of hold constraints in your budget.

Segment Summary

The segment summary describes the budget between one budget pin in a path and the next budget pin in the path. The segment summary shows the input pin and output pin for the segment, along with the path type. In the HTML report, hyperlinks link to the pin summaries and the path summary.

The segment summary also includes the actual circuit delays along the segment, and what portion of the delay is considered to be "fixed". Note that these numbers might not exactly match the numbers of the **report_timing** command. The budget numbers are an aggregate that are used for budget calculations. The number includes timing from all corners, rise and fall delays, assumed clock latency at the launch and capture, and budgeted latch borrowing at the launch and capture.

Next, the summary includes up to three lines that are similar to those in a pin summary. The lines contain "S", "D", and "B" numbers for the portion of the path before the segment (if it exists), the segment, and the portion of the path after the segment (if it exists).

EXAMPLES

The following example summarizes the settings for budget blocks from the **set_budget_options** command.

```
prompt> report_budget -blocks
```

The following example writes pin summaries for the inputs of block1.

```
prompt> report_budget -pins block1/* -input
```

The following example writes all budget information to multiple HTML files in the budget_report directory.

```
prompt> report_budget -html_dir budget_report
```

SEE ALSO

budget_attributes(3)
compute_budget_constraints(2)
create_budget_busplan(2)
get_attribute(2)
get_budgets(2)
report_attributes(2)
set_boundary_budget_constraints(2)
set_budget_options(2)
set_latency_budget_constraints(2)
set_pin_budget_constraints(2)
write_io_constraints(2)
write_script(2)

report_buffer_trees

Reports the buffer tree and its level information at the given driver pin.

SYNTAX

status **report_buffer_trees**
[-from *start_point_list*]
[-depth *max_depth*]
[-connections]
[-hierarchy]
[-physical]

Data Types

start_point_list list
max_depth int

ARGUMENTS

-from *start_point_list*

Specifies the starting point of the buffer tree. The starting point is defined as level 0 of the tree. The *start_point_list* can contain pins, ports and nets.

-depth *max_depth*

Reports only the *max_depth* levels of buffers in the tree. The default is to report the buffer tree until a real load or hierarchical boundary is reached.

-connections

Shows the net connections in the report. By default, the net connections are not shown.

-hierarchy

Shows the buffer tree across the hierarchy. By default, the report terminates at hierarchical boundaries.

-physical

Displays the location for each pin, with the coordinates in microns.

DESCRIPTION

The **report_buffer_trees** command reports a buffer tree at the given driver pins or driver nets. It reports the full name of the driver pin (which includes the name of the cell), the name of the library cell, and the level of the driver relative to the starting point. The starting point is defined as level 0 of the tree.

Reporting terminates at the non-buffer pins, hierarchical boundaries (except when the **-hierarchy** option is used), or when the level reported exceeds the value specified by *max_depth*.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to report a buffer tree:

```
prompt> report_buffer_trees -from cell1/O
```

The following example shows how to report a buffer tree with net connections:

```
prompt> report_buffer_trees -connections -from cell1/O
```

SEE ALSO

create_buffer_trees(2)
remove_buffer_trees(2)

report_bundle_pin_constraints

Reports bundle pin constraints set by the **set_bundle_pin_constraints** command.

SYNTAX

int **report_bundle_pin_constraints**
[-bundles *bundle_collection*]
[-cells *cell_collection*]
[-self]

Data Types

bundle_collection collection
cell_collection collection

ARGUMENTS

-bundles *bundle_collection*

Specifies the net bundles on which the constraints are reported. The bundles must be in the top level of the current design. By default, the command reports all bundles.

-cells *cell_collection*

Specifies the block cells on which the constraints are reported. The cells must be in the top level of the current design. You can specify both the **-cells** and **-self** options. By default, the command reports bundle constraints for all cells if both the **-cells** and **-self** options are omitted.

-self

Reports constraints for the top-level block. You can specify both the **-cells** and **-self** options. By default, the command reports bundle constraints for all cells if both the **-cells** and **-self** options are omitted.

DESCRIPTION

This command reports existing bundle pin constraints. If no options are given, all bundle pin constraints (not include top-level block) in the design are reported.

There are four type of bundle pin constraints. The constraints are listed below from general to specific.

- Global constraints
  Constraint applies to all bundles and all block instances.

- Per-bundle constraints
  Constraint applies to a specific bundle and all block instances.
- Per-cell constraints
  Constraint applies to all bundles and a specific block instance.
- Per-bundle-cell-pair constraints
  Constraint applies to a specific bundle and a specific block instance.

In general, the **report_bundle_pin_constraints** command reports the general constraint and the more specific constraint, if specified. For example, the **report_bundle_pin_constraints -bundles B** command reports both per-bundle constraints that are specific to bundle B and per-bundle-cell-pair constraints that are specific to bundle B and some other cell.

This command returns 1 if some constraints could be reported, 0 otherwise.

EXAMPLES

The following example reports all bundle pin constraints from the design. All four types of constraints are reported.

```
prompt> report_bundle_pin_constraints

****************************************
Report : report_bundle_pin_constraints
 ...

****************************************

Constraints on all cells and bundles
---------------------------------------
Allowed layers:          METAL2 METAL3

Constraints on all cells and bundle(s) B
-------------------------------------------------
Allowed layers:    METAL2 METAL3 METAL4 METAL 5
Bundle order:      increasing
Keep pins together: true

Constraints on cell BLOCK1 and all bundles
--------------------------
Spacing on all layers: 42
Sides:               2 3

Constraints on bundle B2 and cell BLOCK2
-------------------------------------------------
Pin width:  0.1
Pin height: 0.4
```

SEE ALSO

place_pins(2)

remove_bundle_pin_constraints(2)
report_block_pin_constraints(2)
report_individual_pin_constraints(2)
set_bundle_pin_constraints(2)

report_bundles

Report information about the given bundles

SYNTAX

status_value **report_bundles**
bundle_list bundles

Data Types

bundle_list list

ARGUMENTS

bundle_list

Specifies the list of bundles to report information for.

DESCRIPTION

Displays information about the bundle.

EXAMPLES

The following example reports information about a bundle

```
prompt> report_bundles [get_bundles BUNDLE_5]
Reporting bundle {
  Bundle NAME: `  BUNDLE_5`
  Number of objects in bundle 3
  Bundle Type 'net'
  Objects in bundle
     .....
}
1
```

SEE ALSO

add_to_bundle(2)
create_bundle(2)
get_bundles(2)
remove_bundles(2)
remove_from_bundle(2)

report_busplan_constraints

Write a report about busplan constraints.

SYNTAX

int **report_busplan_constraints**
[buses]

Data Types

buses string

ARGUMENTS

buses

Specifies the names of busplans to report. The buses must have been previously created by the **create_busplans** command.

DESCRIPTION

Report about busplans constraints as previously defined by the **set_busplan_constraints** command.

EXAMPLES

Set a constraint for a busplan named bus1 and show a report giving the constraints for busplan bus1.

```
prompt> set_busplan_constraints -from bus1 -to bus2 -type =
prompt> report_busplan_constraints bus1

****************************************
Report : report_busplan_constraints
Design : TOP
Version: L-2016.03-SP2
Date   : Fri Mar 11 19:41:53 2016
****************************************
From bus  Constraint
---------------------------
```

```
bus1        = bus2
1
```

SEE ALSO

create_busplans(2)
set_busplan_constraints(2)
check_busplan_constraints(2)

report_busplans

Writes out a report about busplans.

SYNTAX

int **report_busplans**
[-start_end_cells]
[-location]
[-mib_compliance]
[-consistency]
[-xml_file *filename*]
[buses]

Data Types

filename string
buses string

ARGUMENTS

-start_end_cells

Includes start cells and end cells in the report. These are the cells that buses can start and end at. By default, all hard macros and top-level blocks can be start cells or end cells. You can also specify start cells and end cells with the **create_busplans** command.

-location

Reports the location of the busplan objects.

-mib_compliance

Reports on the MIB compliance of the busplans. MIB compliance is relevant for the busplans that have MIB feedthroughs. This report will show if the busplan has any issues with other busplans using the same feedthroughs in other MIB instances.

-consistency

Reports on the internal consistency of the busplans. This is important with respect to being able to implement the busplans using the given plan. For the timing of the plan to be correct all the bits of the busplan must have the same clock period. Also the timing of the plan depends on the layers being used given by the net estimation rule. So all the bits should have the min_layer and max_layer constraints which overlap the rule. This report will report if any such consistency issue exists which would cause a problem in implementation.

-xml_file *filename*

Creates an XML document which contains a busplan report for the specified busplans. This report is a less detailed version than the XML data saved with *write_busplans -xml_file*, but may be useful for quick overview of the current busplan planning data.

buses

Specifies the names of busplans to report. The buses must be created with the **create_busplans** command.

DESCRIPTION

This command reports busplans that are previously defined with the **create_busplans** command.

In the report, lbb is an abbreviation for launch block budget, cbb is an abbreviation for capture block budget.

EXAMPLES

The following example creates a busplan named bus1 and reports the details for the busplan.

```
prompt> set_net_estimation_rule fast_rule -parameter ns_per_mm -value 0.5
prompt> create_busplans -name bus1 -rule fast_rule -from B1/OUT1[*]
prompt> report_busplans -start_end_cells bus1

****************************************
Report : report_busplans
Design : TOP
Version: L-2016.03-DEV
Date   : Sun Aug  2 16:55:49 2015
****************************************
Start/end at instances: B1 B2 B3 B4
bus1: 8 bits start=B1/OUT1[7] clock=-unknown- lbb=0.00 cbb=0.00
  pin              group1     (  8) B1/OUT1[7]
    register         group2     (  8) r1/r7
      register       group3     (  8) r2/r7
        pin          group4     (  8) B2/IN1[7]
1
```

SEE ALSO

create_busplans(2)
get_busplans(2)
modify_busplan(2)
set_net_estimation_rule(2)
write_busplans(2)

report_case_analysis

Reports case analysis entries on ports and pins.

SYNTAX

string **report_case_analysis**
[-mode *mode*]
[-all]
[-nosplit]

Data Types

mode collection

ARGUMENTS

-mode *mode*

A single mode to be used for the report. This can be a collection or the name of a valid mode. If not specified, the current mode will be used.

-all

Reports the pins upon which you have set case analysis values and reports the built-in constant pins of the design that are considered for start points of logic constant propagation. Logic constant propagation is performed by default from the constant pins of the design, unless the **time.disable_case_analysis** application option is set to **true**.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. If you do not use this option, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line starting in the correct column.

DESCRIPTION

Reports case analysis entries on ports and pins that are specified with the **set_case_analysis** command. The report lists all pins and ports on which case analysis is specified. The list does not report where the logic constant values are propagated.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-mode** option.

EXAMPLES

The following example specifies that pins *U1/U2/A* and *U1/U3/CI* are set to a constant logic *1*.

```
prompt> set_case_analysis 1 {U1/U2/A U1/U3/CI}
prompt> report_case_analysis

****************************************
Report : case_analysis
Module : counter
Mode   : m2
Version:
Date   : Thu Jan  5 19:38:06 2017
****************************************

Pin name                        User case analysis value
--------------------------------------------------------------------------------
U1/U2/A                         1
U1/U3/CI                        1
```

SEE ALSO

remove_case_analysis(2)
set_case_analysis(2)

report_ccd_timing

Reports CCD timing information of a design. It can be used to report D and Q slacks of critical endpoints (D slack is the worst slack of the paths captured by the flop and Q slack is worst slack of the paths launched by the flop) and analyze slacks of timing paths that precede or succeed the critical timing path. It also provides options to prepone and postpone clock pin delays and annotate delays on nets and cells to analyze possible ways to fix timing slacks on critical endpoints.

SYNTAX

string **report_ccd_timing**
[-type] *report_type*
[-max_paths] *max_path_count*
[-max_stages] *max_stage_count*
[-pins] *pin_list*
[-hold]
[-scenarios] *scenario_list*
[-offset] *value*
[-annotate_cell_delay] *pin_pin_value*
[-annotate_net_delay] *pin_pin_value*
[-significant_digits] *digits*
[-nosplit]

Data Types

report_type string
max_path_count integer
max_stage_count integer
pin_list list
scenario_list list
value float
pin_pin_value string
digits integer

ARGUMENTS

-type stage | **chain** | **fanin** | **fanout**

Specifies the type of report to be generated. Type **stage** reports the timing of previous stage, current stage and post stages. Type **chain** shows the timing of whole chain. Type **fanin** shows the timing of fanin cone. Type **fanout** reports timing of fanout cone. If this option is not specified, the default report will be printed. This will report most critical endpoints in the design along with their D and Q slack values.

-max_paths *max_path_count*

Specifies the number of paths to report. The default value is 1.

-max_stages *max_stage_count*

Specifies the number of stages to report. The default value is 1.

-pins *pin_list*

Specifies the pins to work on. If this option combined with -offset together, the -pins should be clock pins.

-hold

When this option is used while reporting the summary and stage reports, it focusses on the most critical hold violations in the design and prints the D and Q slacks for hold for these endpoints and also prints corresponding setup slacks for the hold violating pins. For fanin and fanout reports, the hold slacks of all the pins in the fanin and fanout of the specified pin are reported. This option is useful while debugging hold violations in a design.

-scenarios *scenario_list*

By default, report_ccd_timing analyzes timing paths across all active scenarios and reports the worst D and Q slacks across scenarios. If user wants to analyze a particular scenario (or a set of scenarios) and report slacks specifically from those scenarios, the scenario list can be specified using the -scenarios option.

-offset *value*

Specifies the incremental value advancing/delaying on the pins, and then reports the timing change on the design as a result of the offset. This option requires option -pins.

-annotate_cell_delay *value*

Reports the timing change with specified annotated cell delay.

-annotate_cell_delay *value*

Reports the timing change with specified annotated net delay.

-significant_digits *digits*

Specifies the number of significant digits to show for reporting of any floating point number field. By default, each field has a particular number of significant digits, intended to communicate the right level of information to the user. This option overrides those default settings.

-nosplit

Writes out wide report lines, even if the line length is greater than 80 columns. If certain fields exceed their fixed column width, then printing can automatically jump to the next line to maintain the formatting. This is done to improve readability of the report. If -nosplit is specified, then one row of data in a tabular report will always be printed on one row. The columns will no longer align, which affects readability of the report. The -nosplit option is useful to simplify parsing or post-processing of a report output file. If the report is not intended to be post-processed, then do not use the nosplit option, in order to get better report readability.

DESCRIPTION

The **report_ccd_timing** command serves the following purposes using appropriate switches:

=== Report Stage Timing ===

This command has ability to generate a report that contains timing information for the sub-netlist that user focuses on. The **-type** and **-pin** report can show the timing slacks of previous/next stage/fanin/fanout timing paths for a particular timing end point or timing slacks of the entire chain using **-type** chain. Using this, user can directly check if there is any opportunity to borrow timing from

adjacent stages. (**-max_stages** is provided to control the number of stages to be printed.)

=== Fanin/Fanout reports ===

Report_ccd_timing can print the D and Q slacks of all the pins in the fanin or fanout of a specified pin in the design. By default it prints 5 timing paths in the fanin/fanout cone. To report more paths, **-max_paths** option can be used. This report prints the setup slacks by default. To print hold slacks, use **-hold** option.

=== Offset ===

This command supports some options where user can apply offset delays on specified clock pins and view the resulting timing change on the design. It quickly calculates the WNS and TNS change with the skew adjustment specified by the user. Now we don't support this feature for the design with ideal clock network.

=== Annotated Delay ===

User can use **-annotate_cell_delay**/**-annotate_net_delay** options to adjust cell delay or net delay to check the resulting timing change. This cell delay/net delay adjustment will be reverted automatically after reporting is done. User can easily specify the annotated cell delay or net delay and the reporting command will reflect resulting WNS/TNS change. Hence, user can figure out whether there is room to get timing improvement and also understand the changes to the timing picture with the specified annotated delay. The specified delay value overrides the internally-estimated cell and net delay value.

Note that the annotated delay values and offset settings will be applied to current scenario by default, and this value will be scaled and applied to other scenarios.

EXAMPLES

The following example can generate a summary of slacks on setup critical endpoints.

```
prompt > report_ccd_timing
```

The following example can generate a summary report of the top 3 setup critical paths.

```
prompt > report_ccd_timing -max_paths 3
```

To report the hold critical endpoints, use the -hold option.

```
prompt > report_ccd_timing -hold
```

In the following example, the command shows the pin slacks of the previous stage, current stage an post stage of the pin PC_reg_9/D.

```
prompt > report_ccd_timing -pin PC_reg_9/D -type stage
```

In the following example, the command shows the pin slacks of the whole chain.

```
prompt > report_ccd_timing -pin I2/d -type chain
```

In this example, the command reports 10 endpoints in the fanout of the pin specified by the user and their respective setup slacks.

```
prompt > report_ccd_timing -type fanout -pin sink1/D -max_paths 10
```

In the following example, the command annotates a net delay of 0.066 units between output pin U1/y and input pin I2/clk in scenario s1.

```
prompt > current_scenario <s1>
prompt > report_ccd_timing -annotate_net_delay {{U1/y I2/clk 0.066}}
```

In the following example, the command annotates a cell delay of 0.12 units between input pin U1/a and output pin U1/y of the same cell instance. And -stage option can be used to print timing changes of previous stage/post stage of the pin specified by the user, l2/d.

```
prompt > current_scenario <s1>
prompt > report_ccd_timing -annotate_cell_delay {{U1/a U1/y 0.12}} -pin l2/d -type stage
```

In this example, we delay the clock by 20ps. We can see the overall timing change when this delay is added.

```
prompt> report_ccd_timing -offset {{l1/clk 0.02}}
```

SEE ALSO

report_clock_qor(2)
report_clock_timing(2)
report_timing(2)

report_cell

Reports cell information.

SYNTAX

string **report_cells**
[-connections]
[-pvt]
[-power]
[-verbose]
[-significant_digits *digits*]
[-nosplit]
[*cell_list*]

list *cell_names*

ARGUMENTS

-connections

Displays the pins and the nets to which they connect.

-verbose

Displays verbose connection information. For each input pin, displays the net and the driver pins on that net. For each output pin, displays the net and the load pins on that net. Use this option in conjunction with **-connections** or the **-pvt** options.

-pvt

Displays the current corner's PVT information related to this cell. By default this includes the early and late process number, process label (if any), default voltage, and temperature. If the **-verbose** option is given, more information will be printed describing each parameter's specifed and effective values, and the file and line number of the constraint (if any) that set the parameter.

-power

Displays the power connections for the cell. For each power pin, a line will be printed giving the pin name, the pin type (power, ground, etc.), the power-type (primary, backup, etc.), the library power rail associated with the pin, the net (if any) connected to the pin, the UPF supply_net connected to the pin, and the early and late voltage of the supply_net. A "*" following the supply_net name means that the supply_net is explicitly connected to the power pin by a UPF **connect_supply_net** statement. A "*" following the early or late voltage value means that the supply_net voltage has been explitly set on the supply_net by a **set_voltage** command. Otherwise, the voltage was derived from some other part of a connected supply_net network.

-nosplit

Does not split lines if column overflows.

-significant_digits ***digits***

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, whose default value is 2. Use this option if you want to override the default. Fixed-precision floating-point arithmetics is used for delay calculation; therefore, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h. The upper bound on a meaningful value of the argument to **-significant_digits** is then FLT_DIG - ceil(log10(fabs(*any_reported_value*))).

cell_list

Lists cells to report. If you do not specify this option, all cells in the current instance are listed.

DESCRIPTION

Displays information and statistics about cells in the current instance or current design. If you set for the current instance, the report is generated for the design of that instance. Otherwise, the report is generated for the current design.

Some information, such as whether the cell is in an ideal network or not, is displayed after a timing update (as a result of manually executing an update_timing function or experiencing an implicit timing update as a result of executing other commands). This behavior is intended to help you to obtain information and statistics about cells without incurring the cost of a complete timing update.

EXAMPLES

The following example displays a report of the cells in the current design.

```
prompt> report_cells
****************************************
Report : cell
 ...

****************************************

Attributes:
   b - black-box (unknown)
   h - hierarchical
   n - noncombinational
   u - contains unmapped logic
   E - extracted timing model
   Q - Quick timing model
   U - Unbound (missing)
   s - user size_only
   S - implicit size_only
   d - user dont_touch
   D - derived dont_touch


Cell                Reference      Library          Area  Attributes
------------------------------------------------------------------------------
i1                  inter                        6.00  h
```

```
low_i                low                          2.00  h
n0_i                 ND2          my_lib           1.00
o_reg2               FD2           my_lib          9.00  n
--------------------------------------------------------------------------------
Total 4 cells                                     18.00

1
```

The following example gives a verbose report of the cell connections.

```
prompt> report_cells -verbose -connections
****************************************
Report : cell
 -connections
 -verbose
 ...

****************************************


Connections for cell 'i1':
   Reference:        inter
   Hierarchical:     TRUE
   Area:             6

   Input Pins        Net          Net Driver Pins   Driver Pin Type
   ----------------  -----------  ----------------  ----------------
   a                 out_1        o_reg1/Q          Output Pin (FD2)
   b                 out_2        o_reg2/Q          Output Pin (FD2)
   c                 out_3        o_reg3/Q          Output Pin (FD2)
   d                 out_4        o_reg4/Q          Output Pin (FD2)

   Output Pins       Net          Net Load Pins     Load Pin Type
   ----------------  -----------  ----------------  ----------------
   z1                i1t1         low_i/n1/A        Input Pin (NR4)
   z2                i1t2         low_i/n1/B        Input Pin (NR4)
   z3                i1t3         low_i/n1/C        Input Pin (NR4)

Connections for cell 'low_i':
   Reference:        low
   Hierarchical:     TRUE
   Area:             2

   Input Pins        Net          Net Driver Pins   Driver Pin Type
   ----------------  -----------  ----------------  ----------------
   a                 i1t1         i1/low/n1/Z       Output Pin (NR4)
   b                 i1t2         i1/low2/n1/Z      Output Pin (NR4)
   c                 i1t3         i1/low3/n1/Z      Output Pin (NR4)
   d                 in2          in2               Input Port

   Output Pins       Net          Net Load Pins     Load Pin Type
   ----------------  -----------  ----------------  ----------------
   z                 low          o_reg2/D          Input Pin (FD2)

Connections for cell 'n0_i':
   Reference:        ND2
```

```
    Library:          my_lib
    Area:             1

    Input Pins        Net          Net Driver Pins   Driver Pin Type
    ----------------  -----------  ----------------  ----------------
    A                 p_in         p_in              Input Port
    B                 p_in         p_in              Input Port

    Output Pins       Net          Net Load Pins     Load Pin Type
    ----------------  -----------  ----------------  ----------------
    Z                 n0           n1_i/B            Input Pin (ND2)

  Connections for cell 'o_reg2':
    Reference:        FD2
    Library:          my_lib
    Area:             9

    Input Pins        Net          Net Driver Pins   Driver Pin Type
    ----------------  -----------  ----------------  ----------------
    D                 low          low_i/n1/Z        Output Pin (NR4)
    CP                CLOCK        CLOCK             Input Port
    CD                OPER         OPER              Input Port

    Output Pins       Net          Net Load Pins     Load Pin Type
    ----------------  -----------  ----------------  ----------------
    Q                 out_2        i1/low3/n1/B      Input Pin (NR4)
                                   out_2             Output Port
                                   i1/low/n1/B       Input Pin (NR4)
                                   i1/low2/n1/B      Input Pin (NR4)
    QN                NET2QNX      i2/low3/n1/B      Input Pin (NR4)
                                   i2/low/n1/B       Input Pin (NR4)
                                   i2/low2/n1/B      Input Pin (NR4)

  1
```

The following example gives a report of the cell PVT settings.

```
prompt> report_cells -pvt
****************************************
Report : cell
Design : middle
Version:
Date   :
****************************************
Cell-specific PVT parameters for corner default

Cell                    Process Label   Process Number   Voltage   Temperature   Pane
-------------------------------------------------------------------------------------
n0_i                    early   (none)        1.00         5.00      25.00        0
                        late    (none)        1.00         5.00      25.00        0
n1_i                    early   (none)        1.00         5.00      25.00        0
                        late    (none)        1.00         5.00      25.00        0
n2_i                    early   (none)        1.00         5.00      25.00        0
                        late    (none)        1.00         5.00      25.00        0
n3_i                    early   (none)        1.00         5.00      25.00        0
                        late    (none)        1.00         5.00      25.00        0
o_reg1                  early   (none)        1.00         5.00      25.00        0
```

```
                late    (none)       1.00          5.00     25.00       0
o_reg2              early   (none)       1.00          5.00     25.00       0
                late    (none)       1.00          5.00     25.00       0
o_reg3              early   (none)       1.00          5.00     25.00       0
                late    (none)       1.00          5.00     25.00       0
o_reg4              early   (none)       1.00          5.00     25.00       0
                late    (none)       1.00          5.00     25.00       0

1
```

The following example gives a verbose report of one cell's PVT settings.

```
prompt> report_cells -pvt -verbose o_reg1
****************************************
Report : cell
Design : middle
Version:
Date   :
****************************************


PVT Parameters for cell 'o_reg1':
   Reference:        FD2
   Library:          lsi_10k
   Corner:           default

Parameter                Specified Value    Effective Value    Source            File/line
-------------------------------------------------------------------------------------------------------
early process label        (none)           (none)           --                --
early process number         1.00             1.00             default           --
default early voltage        5.00             5.00             default           --
early temperature           40.00            25.00             explicit setting    --
early pane                        0

late process label         (none)           (none)           --                --
late process number          1.00             1.00             default           --
default late voltage         5.00             5.00             default           --
late temperature            40.00            25.00             explicit setting    --
late pane                         0

1
```

The following example gives a power report for one cell.

```
prompt> report_cells -power or1
****************************************
Report : cell
Design : sub2
Version:
Date   :
****************************************


Power data for cell 'or1':
Corner: default
```

Power pins:

```
  Pin Name      Type     PG Type   Power Rail  Netlist Net    UPF Supply Net Early Voltage  Late Voltage
-------------------------------------------------------------------------------------------------------
  VDD          power    primary   VDD                vdd          1.25          1.08
  VSS          ground   primary   VSS                vss          --            --
1
```

SEE ALSO

current_design(2)
current_instance(2)
report_design(2)
report_hierarchy(2)
report_net(2)
report_port(2)
report_references(2)

report_cell_array_patterns

Reports cell array patterns in the current block.

SYNTAX

int **report_cell_array_patterns**
[-design *design*]
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*cell_array_pattern_list*]

Data Types

design collection
digits int
cell_array_pattern_list collection

ARGUMENTS

-design *design*

Specifies the block for finding cell array patterns. If a design is not specified, the current block is used.

-verbose

Displays verbose cell array pattern information.

-nosplit

Retains long lines and does not split lines if column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13. The default is determined by the **shell.report_default_significant_digits** application option, whose default is 2. Use this option to override the default.

cell_array_pattern_list

Specifies a list of cell array patterns to report. The list can contain cell array pattern names, patterns, or collections. A collection can be specified by using the **get_cell_array_patterns** command.

DESCRIPTION

The **report_cell_array_patterns** command generates a report about the cell array patterns in the design. The report includes the cell array pattern name and a description of the cell array pattern. If **-verbose** is specified, the report also includes additional attributes.

If *cell_array_pattern_list* is specified, those cell array patterns are reported. If it is not specified, then all cell array patterns in the design are reported.

EXAMPLES

The following example reports information for all of the cell_array_patterns in the current block.

```
prompt> report_cell_array_patterns *
Cell Array Pattern Description
--------------------------------------------------------------------------------
pattern0        2 x 3 solid
                lib_cell_name: sysbuf

pattern1        5 x 4 checkerboard
                lib_cell_name: invgate2

1
```

SEE ALSO

get_cell_array_patterns(2)
create_cell_array_pattern(2)
remove_cell_array_patterns(2)

report_cell_buses

Reports bussed cells and contained scalar cells in unmapped or mapped designs.

SYNTAX

report_cell_buses
[-hierarchical]
[-nosplit]
[-expand]
[*object_list*]

ARGUMENTS

-hierarchical

Report bussed cells in all sub-modules of the specified designs (or of the top design if nothing was specified in the *object_list*).

-nosplit

Prevents line splitting. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-expand

Report information on scalar cells of a bussed cell in expanded view. If the **-expand** option is not specified, only bussed cells are reported (collapsed view).

DESCRIPTION

The **report_cell_buses** command lists information about bussed cells. It reports the total number of bussed cells in a design, and information on each bussed cell including: the bus width, the total number of cells that implement a bussed cell, the scalar cells references and corresponding technology libraries, attributes and restrictions set by the user that have an impact on multibit mapping and slice preservation.

When the **-hierarchical** option is specified, the command reports the bussed cells information for all sub-modules of the specified designs.

When the **-expand** option is specified, the report details on each contained scalar instance to provide information on mapping and user restrictions.

The optional [*object_list*] may contain individual cells, bussed cells, and/or designs. When not specified, the command reports the bussed cell information for the top design.

EXAMPLES

The following example generates bussed cell report for the top level of a mapped design in collapsed view:

```
prompt> report_cell_buses

****************************************
Report : report_cell_buses
 ...

****************************************


Restrictions :
   e - excluded from multibit mapping
   d - dont_touch attribute
   s - slice preserved

Report Bussed Cells for design  'KernFilt'
Bussed cell                                 Width    Total Cells  Restrictions
--------------------------------------------------------------------------------------------------
rTu1_reg                                   15       6
rTu2_reg                                   10       4
v_rTuc0_reg                                  25       6
v_rTuc1_reg                                  25       7
--------------------------------------------------------------------------------------------------
Total bussed cells: 4
1
```

The following example generates bussed cell report for and individual bussed cell in expanded view:

```
prompt> report_cell_buses -expand [get_cell_buses rTu2_reg]
****************************************
Report : report_cell_buses
 ...

****************************************


Attributes:
   b - black box (unknown)
   h - hierarchical
   n - noncombinational
   r - removable
   u - contains unmapped logic

Restrictions :
   e - excluded from multibit mapping
   d - dont_touch attribute
   s - slice preserved

Bussed Cell : rTu2_reg
Cell                    Reference     Library      Area  Width  Attributes Restrictions
--------------------------------------------------------------------------------------------------
rTu2_reg[9:6][0]              FJSCC1SDFFC4QRBXC1
```

```
                                TSMC_N28_SP8M5X2ZRDL_TCBN28
                                    10.00 4     n
rTu2_reg[5:2][0]            FJSCC1SDFFC4QRBXC1
                                TSMC_N28_SP8M5X2ZRDL_TCBN28
                                    10.00 4     n
rTu2_reg[1]                 FJSCC1SDFFQRBXA5
                                TSMC_N28_SP8M5X2ZRDL_TCBN28
                                    2.00  1     n
rTu2_reg[0]                 FJSCC1SDFFQRBXA5
                                TSMC_N28_SP8M5X2ZRDL_TCBN28
                                    2.00  1     n
--------------------------------------------------------------------------------------------
Total  4     cells                          24.00 10
1
```

The following example generates expanded and hierarchical bussed cell report for an unmapped design that has user restrictions on bussed cells:

```
prompt> report_cell_buses -hierarchical -expand
****************************************
Report : report_cell_buses
 ...

****************************************


Attributes:
   b - black box (unknown)
   h - hierarchical
   n - noncombinational
   r - removable
   u - contains unmapped logic

Restrictions :
   e - excluded from multibit mapping
   d - dont_touch attribute
   s - slice preserved

Report Bussed Cells for design  'top'
Bussed Cell : U1/out_reg
Cell                    Reference     Library      Area  Width  Attributes Restrictions
--------------------------------------------------------------------------------------------
out_reg[3]                **SEQGEN**    -          0.00  1     n, u
out_reg[2]                **SEQGEN**    -          0.00  1     n, u
out_reg[1]                **SEQGEN**    -          0.00  1     n, u      e
out_reg[0]                **SEQGEN**    -          0.00  1     n, u
--------------------------------------------------------------------------------------------
Total  4     cells                          0.00  4

Bussed Cell : U2/out_reg
Cell                    Reference     Library      Area  Width  Attributes Restrictions
--------------------------------------------------------------------------------------------
out_reg[3]                **SEQGEN**    -          0.00  1     n, u      s
out_reg[2]                **SEQGEN**    -          0.00  1     n, u      s
out_reg[1]                **SEQGEN**    -          0.00  1     n, u      s
out_reg[0]                **SEQGEN**    -          0.00  1     n, u      s
--------------------------------------------------------------------------------------------
Total  4     cells                          0.00  4
```

```
--------------------------------------------------------------------------------
Total bussed cells: 2
1
```

SEE ALSO

compile.multibit.enable(3)
compile(2)
get_cell_buses(2)
get_attribute(2)
set_multibit_options(2)
report_cell(2)
report_multibit_banking(2)

report_cell_em

Reports the cell electromigration maximum toggle rate violations or related information.

SYNTAX

status **report_cell_em**
[-pins *pin_object*|*pin_name*]
[-cells *cell_object*|*cell_name*]

Data Types

pin_object	object
pin_name	string
cell_object	object
cell_name	string

ARGUMENTS

-pins *pin_object*|*pin_name*

Specifies the pin instances for which to report cell electromigration information, whether they violate or not. When you use this option, only the information for the specified pins are reported.

-cells *cell_object*|*cell_name*

Specifies the cell instances for which to report cell electromigration information, whether they violate or not. When you use this option, only the information for the pins of the specified cells are reported.

DESCRIPTION

By default, this command reports all pins with cell electromigration maximum toggle rate violations in the current scenario.

You can use the **-pins** or **-cells** option to report cell electromigration information of specific pins or pins of specific cells, whether they violate or not.

Multicorner-Multimode Support

This command works only on the current scenario.

EXAMPLES

The following command reports all the pins with cell electromigration maximum toggle rate violations for the current scenario:

```
prompt> report_cell_em
```

The following command reports cell electromigration information for a specific pin:

```
prompt> report_cell_em -pin BlkB/ff3/Q
```

The following command reports cell electromigration information for all pins of a specific cell:

```
prompt> report_cell_em -cell BlkB/ff3
```

report_cell_em_profiles

Shows information on which cell EM profiles are available and which library they came from.

SYNTAX

string **report_cell_em_profiles**
library

ARGUMENTS

library

Specifies the reference library to report. This can be either the library name or a collection of a single library.

DESCRIPTION

The **report_cell_em_profiles** command displays the cell EM lifetime_profiles available in the specified library and the source DB files where they came from.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example reports the cell EM lifetime_profiles information for the currently opened library.

```
prompt> report_cell_em_profiles [get_libs]

****************************************
Report : cell_em_profiles
Version: Q-2019.12-SP4-DEV
Date   : Wed Mar 11 02:06:53 2020
****************************************

Cell Em Profile     lib:em012
------------------------------------------------
```

```
lpf_0          : /project/ndm/em/em0.db
lpf_c          : /project/ndm/em/em1.db
lpf_c          : /project/ndm/em/em2.db
lpf_b          : /project/ndm/em/em2.db
1
```

SEE ALSO

get_libs(2)

report_cell_feasible_space

Reports feasible spaces for cell placement.

SYNTAX

int **report_cell_feasible_space**
[-voltage_areas *voltage_area_list*]
[-boundary *{ {{llx lly} {urx ury}} | {{x1 y1} {x2 y2} ...} }*]
[-ignore_filler_references *references*]

Data Types

voltage_area_list list
references list

ARGUMENTS

-voltage_areas *voltage_area_list*

Specifies the voltage areas within which to report feasible site spaces. The input list can be names or collections. When specified, only site spaces within these voltage areas are reported. The options **-voltage_areas** and **-boundary** are mutually exclusive.

-boundary *{ {{llx lly} {urx ury}} | {{x1 y1} {x2 y2} ...} }*

Specifies the boundary in which to report feasible site spaces. Only one rectangle or rectilinear polygon is allowed for this option. When specifying this option, only site spaces inside voltage areas within the specified boundary are reported. The options **-voltage_areas** and **-boundary** are mutually exclusive.

-ignore_filler_references *references*

Specifies the references of filler cells which can be ignored when reporting feasible site spaces. The input list can be names or collections. Unfixed filler cells of the specified references will be ignored and the site spaces occupied by them are treated as free spaces. Other filler cells that not of the specified references are still treated as normal cells and cannot not be ignored.

Fixed filler cells are always treated as normal cells and cannot be ignored even if their references are specified.

DESCRIPTION

This command reports feasible site spaces in which to place cells. Feasible site spaces are place-able site areas which subtracts the occupied areas. Occupied areas include areas that contain existing cells (excluding ECO cells) and areas blocked by placement blockages and macros.

By default, this command reports all feasible site spaces in the design. The **-voltage_areas** and **-boundary** options can be used to report spaces within specified voltage areas or boundary.

The feasible site spaces will be grouped by size (width) in the report. And site spaces that smaller than the existing smallest cells are consider as "less than min width".

This command focus on designs in which all cells are legalized except ECO cells. If there are non-ECO cells that are not aligned with sites, they are ignored as free spaces and are added to the collection **$ecoNonLegalCellsForSpaceCheck** and a warning message is issued. ECO cells are also ignored except those whose eco_change_status are **eco_legalized**.

This command supports multiple site defs, and feasible site spaces are classified by different site defs in the report.

EXAMPLES

The following example reports all feasible site spaces in the design.

```
prompt> report_cell_feasible_space
```

The following example reports all feasible site spaces within the specified voltage areas.

```
prompt> report_cell_feasible_space -voltage_areas {pd2 pd3}
```

The following example reports all feasible site spaces within the specified boundary.

```
prompt> report_cell_feasible_space -boundary {{10 10} {40 40}}
```

The following example shows how to ignore fillers of specified references when reporting feasible site spaces.

```
prompt> set references [get_lib_cells {lib1/FILL1 lib1/FILL2}]
prompt> report_cell_feasible_space -ignore_filler_references $references
```

SEE ALSO

report_eco_physical_changes(2)

report_cell_modes

Displays a report of modes for specified cells.

SYNTAX

string **report_cell_modes**
[-nosplit]
[-missing]
[*cell_list*]

Data Types

cell_list list

ARGUMENTS

-nosplit

Writes report information in a single line without splitting wide lines. Most of the mode information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

-missing

Writes out cells with mode groups with no **set_cell_mode** setting. If this option is given, only those cells with mode groups, but have no **set_cell_mode** specified will be printed.

cell_list

Specifies a list of cells whose modes are to be reported. By default, the report includes all cells for which modes have been set or changed by the user.

DESCRIPTION

This command reports cell modes. When reporting cell modes, the report specifies whether the cell mode is enabled or disabled. There are two types of cells with modes. Those with mode groups defined in the original .lib source of the library and those with cell modes created by the ETM flow of the library manager. For cell modes included in .lib source of the library, by default, all modes are enabled in a given cell unless a **set_cell_mode** command is requesting a specific mode to be activated. For the cell modes added by the ETM flow of the library manager, no modes will be activated unless a **set_cell_mode** command is issued. The report will show the cell, its various modes and whether or not they are enabled, and the possible condition statement from the library. Condition statement never exists for modes created by the library manager ETM flow. It is only possible for one mode per group to

be ENABLED at one time.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example reports cell mode information for a cell in its default state (before any **set_cell_mode** command is applied). No modes are currently active so all modes are enabled.

```
prompt> report_cell_modes [get_cells *]
****************************************
Report : cell_mode
 ...
****************************************

Cell          Mode(Group)        Status      Condition
------------------------------------------------------------------------
etm(des_2)     stdby(etm_modes)    ENABLED     desIn[2]
               all_on(etm_modes)   disabled
------------------------------------------------------------------------
```

The following example shows the status of cell ETM after some mode are activated. Cell ETM is set to operate in mode stdby. All timing arcs associated with mode stdby are enabled. All timing arcs associated with mode all_on are disabled.

```
prompt> set_cell_mode stdby etm
prompt> report_cell_modes
****************************************
Report : cell_mode
 ...
****************************************

Cell          Mode(Group)        Status      Condition
------------------------------------------------------------------------
etm(des_2)     stdby(etm_modes)    ENABLED     desIn[2]
               all_on(etm_modes)   disabled
------------------------------------------------------------------------
```

SEE ALSO

reset_cell_mode(2)
set_cell_mode(2)

report_cell_pin_access

Show details of pin access check in NPLDRC

SYNTAX

status **report_cell_pin_access**
-cells *cells*
[-details]
[-fixed]

Data Types

cells	collection
details	boolean
fixed	boolean

ARGUMENTS

-cells *cells*

Specifies the cell instances to analyze for pin access. This is a required option.

-details

Specifies verbosity level of output. Default is false.

-fixed

Specifies whether fixed cells are to be analyzed.

DESCRIPTION

This command tests a collection of instance cells for their routeability as determined by the NPLDRC pin access check. For each cell int the collection, the report lists the types of access checks performed and whether the cell is considered accessible at its current location and orientation. Option "-details" provides detail information about each pin and its access points that were checked.

EXAMPLES

The following example analyzes a specific cell.

```
prompt> report_cell_pin_placement -cells [get_cell cellA] -detail
```

SEE ALSO

legalize_placement(2)
check_legality(2)
analyze_lib_cell_placement(2)

report_cells

Reports cell information.

SYNTAX

string **report_cells**
[-connections]
[-pvt]
[-power]
[-verbose]
[-significant_digits *digits*]
[-nosplit]
[*cell_list*]

list *cell_names*

ARGUMENTS

-connections

Displays the pins and the nets to which they connect.

-verbose

Displays verbose connection information. For each input pin, displays the net and the driver pins on that net. For each output pin, displays the net and the load pins on that net. Use this option in conjunction with **-connections** or the **-pvt** options.

-pvt

Displays the PVT information related to this cell.

-power

Displays the power connections for the cell.

-nosplit

Does not split lines if column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default. Fixed-precision floating-point arithmetic is used for delay calculation; therefore, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h. The upper bound on a meaningful value of the argument to **-significant_digits** is then FLT_DIG - ceil(log10(fabs(*any_reported_value*))).

cell_list

Lists cells to report. If you do not specify this option, all cells in the current instance are listed.

DESCRIPTION

Displays information and statistics about cells in the current instance or current design. If you set for the current instance, the report is generated for the design of that instance. Otherwise, the report is generated for the current design.

Some information, such as whether the cell is in an ideal network or not, is displayed after a timing update (as a result of manually executing an update_timing function or experiencing an implicit timing update as a result of executing other commands). This behavior is intended to help you to obtain information and statistics about cells without incurring the cost of a complete timing update.

Multicorner-Multimode Support

This command works only on the current corner.

EXAMPLES

The following example displays a report of the cells in the current design.

```
prompt> report_cells
****************************************
Report : cell
 ...

****************************************

Attributes:
   b - black-box (unknown)
   h - hierarchical
   n - noncombinational
   u - contains unmapped logic
   E - extracted timing model
   Q - Quick timing model
   U - Unbound (missing)
   s - user size_only
   S - implicit size_only
   d - user dont_touch
   D - derived dont_touch


Cell                Reference      Library          Area  Attributes
--------------------------------------------------------------------------------
i1                  inter                        6.00  h
low_i                 low                          2.00  h
n0_i                  ND2          my_lib          1.00
o_reg2                FD2            my_lib          9.00  n
--------------------------------------------------------------------------------
```

```
Total 4 cells                                18.00

1
```

The following example gives a verbose report of the cell connections.

```
prompt> report_cells -verbose -connections
****************************************
Report : cell
 -connections
 -verbose
 ...

****************************************


Connections for cell 'i1':
   Reference:        inter
   Hierarchical:     TRUE
   Area:             6

   Input Pins        Net          Net Driver Pins   Driver Pin Type
   ----------------  -----------  ----------------  ----------------
   a                 out_1        o_reg1/Q          Output Pin (FD2)
   b                 out_2        o_reg2/Q          Output Pin (FD2)
   c                 out_3        o_reg3/Q          Output Pin (FD2)
   d                 out_4        o_reg4/Q          Output Pin (FD2)

   Output Pins       Net          Net Load Pins     Load Pin Type
   ----------------  -----------  ----------------  ----------------
   z1                i1t1         low_i/n1/A        Input Pin (NR4)
   z2                i1t2         low_i/n1/B        Input Pin (NR4)
   z3                i1t3         low_i/n1/C        Input Pin (NR4)

Connections for cell 'low_i':
   Reference:        low
   Hierarchical:     TRUE
   Area:             2

   Input Pins        Net          Net Driver Pins   Driver Pin Type
   ----------------  -----------  ----------------  ----------------
   a                 i1t1         i1/low/n1/Z       Output Pin (NR4)
   b                 i1t2         i1/low2/n1/Z      Output Pin (NR4)
   c                 i1t3         i1/low3/n1/Z      Output Pin (NR4)
   d                 in2          in2               Input Port

   Output Pins       Net          Net Load Pins     Load Pin Type
   ----------------  -----------  ----------------  ----------------
   z                 low          o_reg2/D          Input Pin (FD2)

Connections for cell 'n0_i':
   Reference:        ND2
   Library:          my_lib
   Area:             1

   Input Pins        Net          Net Driver Pins   Driver Pin Type
   ----------------  -----------  ----------------  ----------------
```

```
    A               p_in         p_in           Input Port
    B               p_in         p_in           Input Port

    Output Pins       Net          Net Load Pins    Load Pin Type
    ----------------  -----------  ---------------- ----------------
    Z               n0           n1_i/B         Input Pin (ND2)

  Connections for cell 'o_reg2':
    Reference:        FD2
    Library:          my_lib
    Area:             9

    Input Pins        Net          Net Driver Pins   Driver Pin Type
    ----------------  -----------  ---------------- ----------------
    D               low          low_i/n1/Z       Output Pin (NR4)
    CP               CLOCK        CLOCK           Input Port
    CD               OPER         OPER            Input Port

    Output Pins       Net          Net Load Pins    Load Pin Type
    ----------------  -----------  ---------------- ----------------
    Q               out_2        i1/low3/n1/B     Input Pin (NR4)
                                 out_2            Output Port
                                 i1/low/n1/B      Input Pin (NR4)
                                 i1/low2/n1/B     Input Pin (NR4)
    QN               NET2QNX      i2/low3/n1/B     Input Pin (NR4)
                                 i2/low/n1/B      Input Pin (NR4)
                                 i2/low2/n1/B     Input Pin (NR4)

  1
```

The following example gives a report of the cells PVT.

```
prompt> report_cells -pvt
****************************************
Report : cell
 ...

****************************************
Cell-specific PVT parameters for corner default

Cell                        Process Label   Process Number
   Voltage   Temperature   Pane
-------------------------------------------------------------------
n0_i                early   (none)        1.00
   5.00     25.00       0
                    late    (none)        1.00
   5.00     25.00       0
n1_i                early   (none)        1.00
   5.00     25.00       0
                    late    (none)        1.00
   5.00     25.00       0
n2_i                early   (none)        1.00
   5.00     25.00       0
                    late    (none)        1.00
   5.00     25.00       0
n3_i                early   (none)        1.00
   5.00     25.00       0
```

```
                    late    (none)        1.00
    5.00    25.00       0
o_reg1              early   (none)        1.00
    5.00    25.00       0
                    late    (none)        1.00
    5.00    25.00       0
o_reg2              early   (none)        1.00
    5.00    25.00       0
                    late    (none)        1.00
    5.00    25.00       0
o_reg3              early   (none)        1.00
    5.00    25.00       0
                    late    (none)        1.00
    5.00    25.00       0
o_reg4              early   (none)        1.00
    5.00    25.00       0
                    late    (none)        1.00
    5.00    25.00       0

1
```

The following example gives a verbose report of one cell PVT.

```
prompt> report_cells -pvt -verbose o_reg1
****************************************
Report : cell
 ...

****************************************


PVT Parameters for cell 'o_reg1':
   Reference:        FD2
   Library:          lsi_10k
   Corner:           default

Parameter                  Specified Value    Effective Value
Source           File/line
------------------------------------------------------------------
early process label        (none)             (none)
--               --
early process number       1.00               1.00
default          --
default early voltage      5.00               5.00
default          --
early temperature          40.00              25.00
explicit setting   --
early pane                            0

late process label         (none)             (none)
--               --
late process number        1.00               1.00
default          --
default late voltage       5.00               5.00
default          --
late temperature           40.00              25.00
explicit setting   --
```

```
    late pane                       0

    1
```

The following example gives a power report for one cell

```
    prompt> report_cells -power or1
    ****************************************
    Report : cell
     ...

    ****************************************


    Power data for cell 'or1':
    Corner: default


    Power pins:

       Pin Name      Type     PG Type   Power Rail  Netlist Net
    UPF Supply Net Early Voltage  Late Voltage
    ------------------------------------------------------------------------
       __VDD        power    primary   __VDD
    vdd          --          --
       __VSS        ground   primary   __VSS
    vss          --          --
    1
```

SEE ALSO

current_design(2)
current_instance(2)
report_design(2)
report_hierarchy(2)
report_net(2)
report_port(2)
report_references(2)

report_check_design_strategy

This command reports the information related to **user-defined checks**.

SYNTAX

status **report_check_design_strategy**
[-checks names_of_user_defined_checks]

Data Types

names_of_user_defined_checks list

ARGUMENTS

-checks ***names_of_user_defined_checks***

Takes a list of names, of one or more **user-defined checks**. Reporting is done only on these given **user-defined checks**.

DESCRIPTION

See man-page of command **check_design** for definitions of **atomic-check**, **pre-defined check**, **mega-check** and **user-defined check**.

report_check_design_strategy command reports information related to **user-defined checks**. A **user-defined check** is a check created using **create_check_design_strategy** command. By default, without any options specified, this command reports the information related to all **user-defined checks**. If user is interested only in getting information related to some specific **user-defined checks**, then she should use the option **-checks**.

Definition of a check is either an UI command with/without its options OR is a list of **atomic-checks**. This command reports the command used for an atomic-check or a list of atomic-checks for a mega-check.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

(1) In the following example, the user-defined check **my_mv_design** is reported.

```
prompt> create_check_design_strategy -define_check my_mv_design { \
   check_mv_design -voltage_threshold 0.7 }
prompt> report_check_design_strategy -checks my_mv_design

****************************************
Report : check_design_strategy
Design :
 ...
****************************************
Check: mv_design
  Current definition:  { check_mv_design -voltage_threshold 0.7 }
****************************************
1
```

(2) In the following example, mega-check **my_mega_check** is created by the user using **create_check_design_strategy** command.

```
prompt> create_check_design_strategy -define_group my_mega_check { \
   mv_design design_mismatch }
prompt> report_check_design_strategy -checks my_mega_check

****************************************
Report : check_design_strategy
Design :
 ...
****************************************

Group: my_mega_check
   Current definition:  { mv_design design_mismatch }
 Check: mv_design
    Current definition:  { check_mv_design -voltage_threshold 0.7 }
 Check: design_mismatch
    Current definition:  { report_design_mismatch -check_design }
****************************************
1
```

(3) In the following example, 2 new user-defined checks are created.

```
prompt> create_check_design_strategy -define_check my_check1 { check_mv_design }
prompt> create_check_design_strategy -define_check my_check2 { check_legality }
prompt> report_check_design_strategy -checks {my_check1 my_check2}

****************************************
Report : check_design_strategy
Design :
 ...
****************************************
Check: my_check1
  Current definition:  { check_mv_design }

Check: my_check2
  Current definition:  { check_legality }
****************************************
1
```

(4) In the following example, default behavior of the command is shown, where information about all user-defined checks is displayed.

```
prompt> report_check_design_strategy

****************************************
Report : check_design_strategy
Design :
 ...
****************************************
Check: mv_design
  Current definition: { check_mv_design -voltage_threshold 0.7 }

Check: my_check1
  Current definition: { check_mv_design }

Check: my_check2
  Current definition: { check_legality }

Group: my_mega_check
   Current definition: { my_mv_design design_mismatch }
 Check: my_mv_design
    Current definition: { check_mv_design -voltage_threshold 0.7 }
 Check: design_mismatch
    Current definition: { report_design_mismatch -check_design }
****************************************
1
```

SEE ALSO

check_design(2)
create_check_design_strategy(2)
get_design_checks(2)

report_checkpoint_options

Reports the parameters set in the chackpoint system.

SYNTAX

integer **report_checkpoint_options**

Data Types

ARGUMENTS

DESCRIPTION

This command reports the configuration of the checkpoint system, set by the set_checkpoint_options command.

EXAMPLES

The following example reports the options set for the checkpoint system.

```
prompt> report_checkpoint_options
```

SEE ALSO

create_checkpoint_action(2)
remove_checkpoint_actions(2)
create_checkpoint_report(2)
remove_checkpoint_reports(2)
associate_checkpoint_action(2)
associate_checkpoint_report(2)
get_current_checkpoint(2)

set_checkpoint_options(2)
eval_checkpoint(2)
get_checkpoint_data(2)

report_clock

Reports clock-related information.

SYNTAX

string **report_clocks**
[-attributes]
[-skew]
[-groups]
[-nosplit]
[-significant_digits *digits*]
[-modes *mode_list*]
[*clock_list*]

Data Types

digits int
mode_list list
clock_list list

ARGUMENTS

-attributes

Shows clock attributes and provides a list of all the clocks in the current design. The information for each clock includes source type, signal rise and fall times, and attributes. This report is shown by default.

-skew

Reports clock latency (source and network latency) and uncertainty information set by the **set_clock_latency** and **set_clock_uncertainty** commands, respectively.

Clock network latency information includes rise latency and fall latency. Clock source latency information includes rise latency and fall latency for early and late arrivals. Clock uncertainty information includes intraclock setup or hold uncertainty and interclock setup or hold uncertainty. This option also reports any fixed clock transition set by using the **set_clock_transition** command.

-groups

Shows the current setting for clock groups, including grouping of exclusive clocks and asynchronous clocks set by using the **set_clock_groups** command.

-nosplit

Retains long lines in the output, even if a column overflows. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This

option prevents line-splitting and facilitates writing scripts to extract information from the report output.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option to override the default.

This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, the tool uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

-modes *mode_list*

Specifies the list of modes to be used in the report. A subreport is produced for each clock in the mode when the *mode_list* is specified.

clock_list

Lists the clocks that must be reported. Substitute the list you want for *clock_names*.

DESCRIPTION

Reports clock-related information. Displays all clock-related information on a design. The report contents are controlled by the options used. If you do not specify any options, the command displays the attributes report.

When a clock is created, it is added to the other clocks list if some clocks are defined to be exclusive or asynchronous with any other clocks in the design. When a clock is removed, it is removed from its related lists. To see if a clock is active or not, use the **report_clocks** command with the **-attributes** option.

Some information, such as the waveform of a generated clock or propagated skew, is displayed after a timing update. This can result from manually running **update_timing** or an implicit timing update as a result of executing other commands. This behavior is intended to help you to define all clocks without incurring the cost of a complete timing update.

To obtain a list of all clocks in the design, use the **all_clocks** command.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-modes** option.

EXAMPLES

```
prompt> create_clock -period 20.000000 \
[get_ports {CLK}]

prompt> report_clocks

****************************************
Report : clock
Design : counter
Version:
Date   :
****************************************
```

```
Attributes:
   p - propagated_clock
   G - Generated clock

Clock        Period   Waveform          Attrs     Sources
-------------------------------------------------------------------------------
CLK          20.00   {0 10}                    {CLK}

prompt> set_clock_latency 2.4 [get_clocks CLK]
prompt> set_clock_latency 0.17 -source -early [get_clocks CLK]
prompt> set_clock_latency 0.19 -source -late  [get_clocks CLK]
prompt> set_clock_uncertainty 1.3 [get_clocks CLK]
prompt> set_clock_transition  1.7 [get_clocks CLK]
prompt> report_clocks -skew

****************************************
Report : clock_skew
Design : counter
Version:
Date   :
****************************************

            Rise         Fall         Hold         Setup
Object       Delay        Delay       Uncertainty    Uncertainty
-------------------------------------------------------------------------------
CLK          2.40         2.40         1.3          1.3

-------------------------------------------------------------------------------
                        Source Latency
-------------------------------------------------------------------------------
Object      Min Rise      Min Fall      Max Rise      Max Fall
-------------------------------------------------------------------------------
CLK          0.17         0.17         0.19          0.19

            Rise         Fall
Object       Transition    Transition
-------------------------------------------------------------------------------
CLK          1.7          1.7

prompt> set_clock_latency -late -source -dynamic 0.5 4.5 [get_clocks clk]
prompt> set_clock_latency -early -source 2.5 -dynamic 0.5 [get_clocks clk]
prompt> report_clocks -skew

****************************************
Report : clock_skew
Design : latency
Version:
Date   :
****************************************

             Min Condition Source Latency      Max Condition Source Latency
-------------------------------------------------------------------------------------------
Object       Early_r Early_f  Late_r  Late_f   Early_r Early_f  Late_r  Late_f  Rel_clk
-------------------------------------------------------------------------------------------
clk (static)    2.00    2.00    4.00    4.00     2.00    2.00    4.00    4.00  --
clk (dynamic)   0.50    0.50    0.50    0.50     0.50    0.50    0.50    0.50  --
           -------------------------------------------------------------------------
```

```
clk (total)     2.50   2.50   4.50   4.50     2.50   2.50   4.50   4.50  --

prompt> set_clock_groups -ex -group {clk1 clk3}
prompt> set_clock_groups -asyn -name a1 -group clk2 -group clk2
prompt> set_active_clocks {clk1 clk2}
prompt> report_clocks -groups

****************************************
Report : clock_groups
Design : test
Version:
Date   : Thu May  9 13:13:51 2002
****************************************

Active clocks:
  clk1  clk2

Total exclusive groups: 1.
NAME : clk1_others
        -group {clk1 clk3 }
        -group {clk2 clk4 }

Total asynchronous groups: 1.
NAME : a1
        -group {clk2 }
        -group {clk4 }
```

SEE ALSO

all_clocks(2)
create_clock(2)
remove_clock_groups(2)
set_clock_groups(2)
set_clock_latency(2)
set_clock_transition(2)
set_clock_uncertainty(2)
shell.common.report_default_significant_digits(3)

report_clock_balance_groups

Reports all stored clock balance groups.

SYNTAX

string **report_clock_balance_groups**
[*clock_balance_group_list*]

Data Types

clock_balance_group_list collection

ARGUMENTS

clock_balance_group_list

Specifies a list of clock balance groups.

DESCRIPTION

This command reports clock balance groups.

EXAMPLES

The following example reports all clock balance groups.

prompt> **report_clock_balance_groups**

SEE ALSO

balance_clock_groups(2)
create_clock_balance_group(2)
get_clock_balance_groups(2)
remove_clock_balance_groups(2)

report_clock_balance_points

Shows the user-defined clock tree balance and exclude points for the current mode.

SYNTAX

int **report_clock_balance_points**
[-clock *clock_list*]
[-balance_points *pin_port_list*]
[-significant_digits *digits*]
[-nosplit]

Data Types

clock_list list
pin_port_list list
digits integer

ARGUMENTS

-clock *clock_list*

Specifies the list of clocks for which to report balance and exclude points. Since balance and exclude points can only be specified on non-generated clock, this report also accepts only non-generated clocks. When no clocks are specified then all the clocks of the current mode are considered.

-balance_points *pin_port_list*

Specifies a list of pins or ports for which to report. If no objects are specified, all the objects for which balance/exclude are specified for the given set of clocks are reported.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 through 13. The default is 2.

-nosplit

Retains long lines in the output, even if a column overflows. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

DESCRIPTION

The **report_clock_balance_points** command generates a subreport for each of the clock specified in the command. Each subreport contains a separate table for balance and exclude points.

Multicorner-Multimode Support

This command works on the current mode.

EXAMPLES

The following example reports all user-defined balance and exclude point information.

```
prompt> report_clock_balance_points
****************************************
Report : clock tree balance points
Module : clock_network1
Mode   : default
****************************************
Balance points for clock 'clk1':
                          Early-   Early-    Late-    Late-
Point                       Rise     Fall     Rise     Rise
--------------------------------------------------------------------------------
gck/CP                      2.00     0.00     0.00     0.00

Exclude points for clock 'clk1':
Point
--------------------------------------------------------------------------------
mux/A

Balance points for clock 'clk2':
                          Early-   Early-    Late-    Late-
Point                       Rise     Fall     Rise     Rise
--------------------------------------------------------------------------------
gck/CP                      0.00     0.00     0.00     0.00

Exclude points for clock 'clk2':
Point
--------------------------------------------------------------------------------
```

SEE ALSO

remove_clock_balance_points(2)
set_clock_balance_points(2)

report_clock_cell_spacings

Reports existing clock cell spacing rules.

SYNTAX

status **report_clock_cell_spacings**
[-clocks *clock_list*]

Data Types

clock_list collection

ARGUMENTS

-clocks *clock_list*

Specifies the clock trees for which the cell spacing settings will be reported.

DESCRIPTION

This command reports all existing clock cell spacing rules,previously defined by **set_clock_cell_spacing** commands. If **-clocks** option is used then it will report cell spacing settings for specified clock networks.

EXAMPLES

The following example reports existing clock cell spacing rules.

prompt> **report_clock_cell_spacings**

SEE ALSO

set_clock_cell_spacing(2)
remove_clock_cell_spacings(2)
cts.placement.cell_spacing_rule_style(3)

report_clock_gate_latency

Reports the details and summary of settings made with **set_clock_gate_latency** command.

SYNTAX

status **report_clock_gate_latency**
[-scenario *scenario_list*]
[-clock *clock*]
[-verbose]
[-nosplit]

Data Types

scenario_list list
clock collection

ARGUMENTS

-scenario *scenario_list*

Reports the clock gate latency settings for all specified scenarios in the list. If you do not specify this option, the report will be generated only for the current scenario.

-clock *clock*

Reports the clock gate latency settings for the given clock in all specified scenarios that have a clock with the same name. All other clocks with different name will be skipped in the report. If you do not specify this option, the report will be generated for all clocks in the specified scenarios.

-verbose

Reports a detailed description of annotated latency values that are inconsistent or missing latency values in clock gates and registers of the design.

-nosplit

Retains long lines in the output, even if a column overflows. Most of the information related with clock gate latency settings is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing scripts to extract information from the report output.

DESCRIPTION

This command will report the settings that were made with **set_clock_gate_latency** command. For each specified scenario, the report will generate a table with the following columns:

- The clock object.
- The stage number.
- The list of latency tuples. Each latency tuple consists of a fan-out range and a latency value

For each clock object, its respective stages will be printed out with their list of latency tuples in a different row. If a clock or scenario has no information related to clock gate latency, it will not be printed.

At the end of the report, a summary of inconsistent and missing latency values will be reported. An inconsistent latency value is found when a gated register or gated clock gate has an annotated latency value which is strictly lower than the annotated latency value of its gating cell. A missing latency value (or gap) is found when a register or clock gate has no annotated latency value.

If the **-verbose** option is used, a detailed list of inconsistencies and gaps will be reported after all tables have been printed.

To remove annotated latencies that were specified using the **set_clock_gate_latency** command, use the **reset_clock_gate_latency** command.

To remove the clock gate latency settings that were specified by the **set_clock_gate_latency** command in the current scenario, use the **set_clock_gate_latency -reset** command.

EXAMPLES

The following four examples will use the following settings:

```
prompt> create_clock clk1 -period 0.5
prompt> create_clock clk2 -period 0.4
prompt> set_clock_gate_latency -stage 0 -fanout_latency {{1-inf -6.02e-2}}
prompt> set_clock_gate_latency -clock clk1 -stage 1 \
  -fanout_latency {{1-inf 1.0}}
prompt> set_clock_gate_latency -clock clk1 -stage 2 \
  -fanout_latency {{1-1 0.1} {2-2 0.2} {3-3 0.3} \
                  {4-4 0.4} {5-5 0.5} {6-6 0.6} \
                  {7-7 0.7} {8-8 0.8} {9-9 0.9} \
                  {10-10 1.0} {11-11 1.1} {12-12 1.2} \
                  {13-13 1.3} {14-inf 1.4}}
```

The first example shows a generated report with the **report_clock_gate_latency** command without any option in the default scenario.

```
prompt> report_clock_gate_latency

****************************************
Report : clock_gate_latency
 ...

****************************************

Scenario:  default

Clock Domain  |  Stage  |  Latency Table
==============|=========|=================
```

```
clk1        | 0     | 1-inf -0.0602
            | 1     | 1-inf 1
            | 2     | 1-1 0.1, 2-2 0.2, 3-3 0.3, 4-4 0.4, 5-5 0.5,
            |       | 6-6 0.6, 7-7 0.7, 8-8 0.8, 9-9 0.9, 10-10 1,
            |       | 11-11 1.1, 12-12 1.2, 13-13 1.3, 14-inf 1.4
--------------+---------+-----------------
clk2        | 0     | 1-inf -0.0602
--------------+---------+-----------------

Warning Summary:

(warning) There are 0 clock latency annotation conflicts. For a full list of
messages use report_clock_gate_latency command with -verbose option

(warning) There are 1 clock latency annotation gaps. For a full list of
messages use report_clock_gate_latency command with -verbose option
```

The second example shows a generated report with the **report_clock_gate_latency** command with the **-clock** option.

```
prompt> report_clock_gate_latency -clock clk1

****************************************
Report : clock_gate_latency
 ...

****************************************

Scenario:  default

Clock Domain  |  Stage  |  Latency Table
==============|=========|=================
clk1        | 0     | 1-inf -0.0602
            | 1     | 1-inf 1
            | 2     | 1-1 0.1, 2-2 0.2, 3-3 0.3, 4-4 0.4, 5-5 0.5,
            |       | 6-6 0.6, 7-7 0.7, 8-8 0.8, 9-9 0.9, 10-10 1,
            |       | 11-11 1.1, 12-12 1.2, 13-13 1.3, 14-inf 1.4
--------------+---------+-----------------

Warning Summary:

(warning) There are 0 clock latency annotation conflicts. For a full list of
messages use report_clock_gate_latency command with -verbose option

(warning) There are 0 clock latency annotation gaps. For a full list of
messages use report_clock_gate_latency command with -verbose option
```

The third example shows a generated report with the **report_clock_gate_latency** command with the **-nosplit** option.

```
prompt> report_clock_gate_latency -nosplit

****************************************
Report : clock_gate_latency
 ...

****************************************

Scenario:  default
```

```
Clock Domain  | Stage  | Latency Table
==============|========|=================
clk1          | 0      | 1-inf -0.0602
              | 1      | 1-inf 1
              | 2      | 1-1 0.1, 2-2 0.2, 3-3 0.3, 4-4 0.4, 5-5 0.5,
                         6-6 0.6, 7-7 0.7, 8-8 0.8, 9-9 0.9, 10-10 1,
                         11-11 1.1, 12-12 1.2, 13-13 1.3, 14-inf 1.4
--------------+---------+-----------------
clk2          | 0      | 1-inf -0.0602
--------------+---------+-----------------

Warning Summary:

(warning) There are 0 clock latency annotation conflicts. For a full list
of messages use report_clock_gate_latency command with -verbose option

(warning) There are 1 clock latency annotation gaps. For a full list of
messages use report_clock_gate_latency command with -verbose option
```

The fourth example shows a generated report with the **report_clock_gate_latency** command with the **-verbose** option.

```
****************************************
Report : clock_gate_latency
Design : top
Version: 2.75.0
Date   : Fri Jun 30 13:15:27 2017
****************************************

Scenario:  default

Clock Domain  | Stage  | Latency Table
==============|========|=================
clk1          | 0      | 1-inf -0.0602
              | 1      | 1-inf 1
              | 2      | 1-1 0.1, 2-2 0.2, 3-3 0.3, 4-4 0.4, 5-5 0.5,
              |        | 6-6 0.6, 7-7 0.7, 8-8 0.8, 9-9 0.9, 10-10 1,
              |        | 11-11 1.1, 12-12 1.2, 13-13 1.3, 14-inf 1.4
--------------+---------+-----------------
clk2          | 0      | 1-inf -0.0602
--------------+---------+-----------------

In these clock domains these pairs of objects have conflicting latency
constraints:

No annnotation conflicts found.


In these clock domains there are latency value gaps:
 Scenario: default. Clock domain: clk2, on stage 1 for clock_gate_out_reg_3.
```

Now, the next three examples will use the following setting instead for three already existing scenarios sc1, sc2 and sc3:

```
prompt> current_scenario sc1
prompt> create_clock clk1 -period 1.5
prompt> create_clock clk2 -period 1.6
prompt> set_clock_gate_latency -stage 1 -fanout_latency {{1-inf 1.7}}
```

```
prompt> set_clock_gate_latency -stage 2 \
  -fanout_latency {{1-10 0.7} {11-inf 0.2}}

prompt> current_scenario sc2
prompt> create_clock clk1 -period 1.7
prompt> set_clock_gate_latency -stage 0 -fanout_latency {{1-inf 1.7}}
prompt> set_clock_gate_latency -stage 1 -fanout_latency {{1-inf 0.3}}

prompt> current_scenario sc3
prompt> create_clock clk2 -period 2.0
prompt> set_clock_gate_latency -stage 0 -fanout_latency {{1-inf 1.5}}
```

The fifth example shows a generated report with the **report_clock_gate_latency** command without any options for the current scenario sc1.

```
prompt> current_scenario sc1
prompt> report_clock_gate_latency

****************************************
Report : clock_gate_latency
 ...

****************************************

Scenario:  sc1

Clock Domain  |  Stage  |  Latency Table
==============|=========|=================
clk1          |  1      |  1-inf 1.7
              |  2      |  1-10 0.7, 11-inf 0.2
--------------+---------+-----------------
clk2          |  1      |  1-inf 1.7
              |  2      |  1-10 0.7, 11-inf 0.2
--------------+---------+-----------------

Warning Summary:

(warning) There are 0 clock latency annotation conflicts. For a full list
of messages use report_clock_gate_latency command with -verbose option

(warning) There are 2 clock latency annotation gaps. For a full list of
messages use report_clock_gate_latency command with -verbose option
```

The sixth example shows a generated report with the **report_clock_gate_latency** command with the **-scenario** option.

```
prompt> current_scenario sc1
prompt> report_clock_gate_latency -scenario [all_scenarios]

****************************************
Report : clock_gate_latency
 ...

****************************************

Scenario:  sc1

Clock Domain  |  Stage  |  Latency Table
```

```
==============|=========|=================
clk1          | 1       | 1-inf 1.7
              | 2       | 1-10 0.7, 11-inf 0.2
--------------+---------+-----------------
clk2          | 1       | 1-inf 1.7
              | 2       | 1-10 0.7, 11-inf 0.2
--------------+---------+-----------------

Scenario: sc2

Clock Domain  | Stage   | Latency Table
==============|=========|=================
clk1          | 0       | 1-inf 1.7
              | 1       | 1-inf 0.3
--------------+---------+-----------------

Scenario: sc3

Clock Domain  | Stage   | Latency Table
==============|=========|=================
clk2          | 0       | 1-inf 1.5
--------------+---------+-----------------

Warning Summary:

(warning) There are 0 clock latency annotation conflicts. For a full list
of messages use report_clock_gate_latency command with -verbose option

(warning) There are 3 clock latency annotation gaps. For a full list of
messages use report_clock_gate_latency command with -verbose option
```

The seventh example shows a generated report with the **report_clock_gate_latency** command with the **-scenario** option and **-clock** option.

```
prompt> current_scenario sc1
prompt> report_clock_gate_latency -scenario [all_scenarios] \
  -clock [get_clocks clk1]

****************************************
Report : clock_gate_latency
 ...

****************************************

Scenario: sc1

Clock Domain  | Stage   | Latency Table
==============|=========|=================
clk1          | 1       | 1-inf 1.7
              | 2       | 1-10 0.7, 11-inf 0.2
--------------+---------+-----------------

Scenario: sc2

Clock Domain  | Stage   | Latency Table
==============|=========|=================
clk1          | 0       | 1-inf 1.7
```

```
              | 1     | 1-inf 0.3
--------------+---------+-----------------
```

Warning Summary:

(warning) There are 0 clock latency annotation conflicts. For a full list
of messages use report_clock_gate_latency command with -verbose option

(warning) There are 1 clock latency annotation gaps. For a full list of
messages use report_clock_gate_latency command with -verbose option

SEE ALSO

apply_clock_gate_latency(2)
compile.clockgate.physically_aware(3)
remove_clock_latency(2)
reset_clock_gate_latency(2)
set_clock_gate_latency(2)
set_clock_latency(2)

report_clock_gating

Reports the details and summary of clock gating in the design.

SYNTAX

status **report_clock_gating**
[-gating_elements]
[-gated]
[-ungated]
[-type {*clock_gate* | *self_gate*}]
[-origin {*tool_inserted* | *pre_existing*}]

ARGUMENTS

-gating_elements

Reports the names of all clock-gating elements, along with their reference names, connected nets, level and whether they have been inserted automatically by the tool (Tool-Inserted), or manually by the user (Pre-Existing).

-gated

Reports the names of all clock gated sequential elements, along with the names of the corresponding clock-gating elements, level and whether they have been inserted automatically by the tool (Tool-Inserted), or manually by the user (Pre-Existing).

-ungated

Reports the names of all registers that are ungated or gated directly by manual inserted clock gates, and the reason why the tool didn't gate them.

-type {*clock_gate* | *self_gate*}

Specifies the type of the gate cells to be reported when using -gated or -gating_elements. The type can be *clock_gate*, *self_gate*, or both. Notice that regardless of the value passed to the **-type** option, if there are self gates in the current design, the self gating summary will be shown.

-origin {*tool_inserted* | *pre_existing*}

Specifies the origin of the clock gates to be reported. The origin can be *tool_inserted*, *pre_existing*, or both. When using *tool_inserted* combined with **-gating_elements** or **-gated** options, those reports are limited to the clock gates inserted by the tool. The summary report still includes information about all clock gates. When using *pre_existing* combined with **-gating_elements** or **-gated** options, those reports are limited to the clock gates inserted manually by the user. The summary report still includes information about all clock gates. If *tool_inserted* and *pre_existing* are used together, or neither are used, every report will consider all clock gates.

DESCRIPTION

This command reports information about clock-gating cells and gated/ungated registers in the current design.

Irrespective of the use of any option, the command always prints at the end a summary report, including this information:

- The total number of clock gates in the design.
- The number and percentage of Tool-Inserted clock gates.
- The number and percentage of Pre-Existing clock gates.
- The number and percentage of gated registers.
- The number and percentage of registers gated by Tool-Inserted clock gates.
- The number and percentage of registers gated by Pre-Existing clock gates.
- The number and percentage of ungated registers.
- The total number of registers.
- The maximum number of clock gate levels.
- The number of multi-level clock gates.

If the design contains multibit registers, the summary report also includes information about bitwidth and multibit composition for the gated/ungated registers.

If the design contains self gates, the summary report also includes an aditional report with the following information:

- The total number of self gates in the design.
- The number and percentage of self gated registers.
- The number and percentage of not self gated registers.
- The total number of registers.

The clock gate levels are counted from clock source to registers, and only clock gates, buffers and inverters are traversed for purposes of counting. Other cells found in the clock path will reset the level counting applicable to the clock gates and their fan-out.

IDENTIFICATION

Clock gates inserted by the tool are named according to the following pattern:

clock_gate_<bank_name>_<number>

The string <bank_name> tipically will match the name of some of the gated registers, and the <number> at the end is used to prevent name collisions. Clock gates whose name has a similar pattern will be recognized as Tool-Inserted clock gates, otherwise they will be recognized as Pre-Existing.

Note that it is possible that clock gates inserted by the user might be recognized unexpectedly as Tool-Inserted, instead of Pre-Existing.

EXAMPLES

The following example shows a summary report generated with the **report_clock_gating** command issued without any option.

```
prompt> report_clock_gating

****************************************
Report : clock_gating
 ...
```

```
****************************************


                Clock Gating Summary
  --------------------------------------------------------------------
  |  Number of Clock gating elements       |          2      |
  |                                        |                 |
  |  Number of Tool-Inserted Clock gates   |       1  (50.00%) |
  |                                        |                 |
  |  Number of Pre-Existing Clock gates    |       1  (50.00%) |
  |                                        |                 |
  |  Number of Gated registers             |       8  (80.00%) |
  |                                        |                 |
  |  Number of Tool-Inserted Gated registers |       4  (40.00%) |
  |                                        |                 |
  |  Number of Pre-Existing Gated registers  |       4  (40.00%) |
  |                                        |                 |
  |  Number of Ungated registers           |       2  (20.00%) |
  |                                        |                 |
  |  Total number of registers             |         10      |
  |                                        |                 |
  |  Max number of Clock Gate Levels       |          1      |
  |                                        |                 |
  |  Number of Multi Level Clock Gates     |          0      |
  --------------------------------------------------------------------
```

This example shows the report generated with the **-gating_elements** option.

```
prompt> report_clock_gating -gating_elements

****************************************
Report : clock_gating
 ...

****************************************


--------------------------------------------------------------------------------
                        Clock Gating Cell Report
--------------------------------------------------------------------------------

 Clock Gating Bank : clock_gate_2 (Level 1) (TLATNTSCAX2MTL) (Pre-Existing)

    INPUTS :
    [CLK  ] clock_gate_2/CK = clk
    [EN   ] clock_gate_2/E = gated_clk1
    [TE   ] clock_gate_2/SE = *Logic0*

    OUTPUTS :
    [ENCLK] clock_gate_2/ECK = gated_clk2
--------------------------------------------------------------------------------
 Clock Gating Bank : clock_gate_out_reg_1 (Level 1) (TLATNTSCAX2MTL) (Tool-Inserted)

    INPUTS :
    [CLK  ] clock_gate_out_reg_1/CK = clk
    [EN   ] clock_gate_out_reg_1/E = en[0]
```

```
    [TE   ] clock_gate_out_reg_1/SE = *Logic0*

    OUTPUTS :
    [ENCLK] clock_gate_out_reg_1/ECK = gated_clk1
-------------------------------------------------------------------------------


                    Clock Gating Summary
    ------------------------------------------------------------------
    |    Number of Clock gating elements        |          2      |
    |                                           |                 |
    |    Number of Tool-Inserted Clock gates    |       1  (50.00%) |
    |                                           |                 |
    |    Number of Pre-Existing Clock gates     |       1  (50.00%) |
    |                                           |                 |
    |    Number of Gated registers              |       8  (80.00%) |
    |                                           |                 |
    |    Number of Tool-Inserted Gated registers |       4  (40.00%) |
    |                                           |                 |
    |    Number of Pre-Existing Gated registers  |       4  (40.00%) |
    |                                           |                 |
    |    Number of Ungated registers            |       2  (20.00%) |
    |                                           |                 |
    |    Total number of registers              |         10      |
    |                                           |                 |
    |    Max number of Clock Gate Levels        |          1      |
    |                                           |                 |
    |    Number of Multi Level Clock Gates      |          0      |
    ------------------------------------------------------------------
```

This example shows the report generated with the **-gated** option.

```
prompt> report_clock_gating -gated

****************************************
Report : clock_gating
 ...

****************************************


-------------------------------------------------------------------------------
                    Gated Register Report
-------------------------------------------------------------------------------
        Clock Gating Bank                |       Gated Register
-------------------------------------------------------------------------------
                                         |
clock_gate_2 (Level 1) (Pre-Existing)    | out_reg[4]
                                         | out_reg[5]
                                         | out_reg[6]
                                         | out_reg[7]
-------------------------------------------------------------------------------
 Total                                                 4
-------------------------------------------------------------------------------
                                         |
clock_gate_out_reg_1 (Level 1) (Tool-Inserted)  | out_reg[0]
                                         | out_reg[1]
```

```
                                | out_reg[2]
                                | out_reg[3]
-------------------------------------------------------------------------------
 Total                                          4
-------------------------------------------------------------------------------
 Sum Total                        2 |                  8
-------------------------------------------------------------------------------


                  Clock Gating Summary
        ------------------------------------------------------------
        |   Number of Clock gating elements        |          2      |
        |                                          |                 |
        |   Number of Tool-Inserted Clock gates    |       1  (50.00%) |
        |                                          |                 |
        |   Number of Pre-Existing Clock gates     |       1  (50.00%) |
        |                                          |                 |
        |   Number of Gated registers              |       8  (80.00%) |
        |                                          |                 |
        |   Number of Tool-Inserted Gated registers |       4  (40.00%) |
        |                                          |                 |
        |   Number of Pre-Existing Gated registers  |       4  (40.00%) |
        |                                          |                 |
        |   Number of Ungated registers            |       2  (20.00%) |
        |                                          |                 |
        |   Total number of registers              |         10      |
        |                                          |                 |
        |   Max number of Clock Gate Levels        |          1      |
        |                                          |                 |
        |   Number of Multi Level Clock Gates      |          0      |
        ------------------------------------------------------------
```

This example shows the report generated with the **-ungated** option.

```
prompt> report_clock_gating -ungated

****************************************
Report : clock_gating
 ...

****************************************



-------------------------------------------------------------------------------
                    Ungated Registers
-------------------------------------------------------------------------------
      Register       |  Reason for not gating   |     What's Next?
-------------------------------------------------------------------------------
 out_reg[8]           | Minimum Bitwidth Not Met   | Check -minimum_bitwidth
                                                     option in set_clock_gating_options.
 out_reg[9]           | Minimum Bitwidth Not Met   | Check -minimum_bitwidth
                                                     option in set_clock_gating_options.
-------------------------------------------------------------------------------
 Total                        2
-------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
          Registers only gated by pre-instantiated clock gates
--------------------------------------------------------------------------------
      Register       |  Reason for not gating   |     What's Next?
--------------------------------------------------------------------------------
out_reg[4]           | Enable is Constant One    | Check RTL for feedback
                                                   loop or synchronous
                                                   enable signal of the register.
out_reg[5]           | Enable is Constant One    | Check RTL for feedback
                                                   loop or synchronous
                                                   enable signal of the register.
out_reg[6]           | Enable is Constant One    | Check RTL for feedback
                                                   loop or synchronous
                                                   enable signal of the register.
out_reg[7]           | Enable is Constant One    | Check RTL for feedback
                                                   loop or synchronous
                                                   enable signal of the register.
--------------------------------------------------------------------------------
 Total                          4
--------------------------------------------------------------------------------



                Clock Gating Summary
        ------------------------------------------------------------
        |  Number of Clock gating elements       |          2      |
        |                                        |                 |
        |  Number of Tool-Inserted Clock gates   |       1  (50.00%) |
        |                                        |                 |
        |  Number of Pre-Existing Clock gates    |       1  (50.00%) |
        |                                        |                 |
        |  Number of Gated registers             |       8  (80.00%) |
        |                                        |                 |
        |  Number of Tool-Inserted Gated registers |       4  (40.00%) |
        |                                        |                 |
        |  Number of Pre-Existing Gated registers  |       4  (40.00%) |
        |                                        |                 |
        |  Number of Ungated registers           |       2  (20.00%) |
        |                                        |                 |
        |  Total number of registers             |         10      |
        |                                        |                 |
        |  Max number of Clock Gate Levels       |          1      |
        |                                        |                 |
        |  Number of Multi Level Clock Gates     |          0      |
        ------------------------------------------------------------
```

This example shows the report generated with the **-gated** option in combination with the **-origin {pre_existing}** option.

```
prompt> report_clock_gating -gated -origin {pre_existing}

****************************************
Report : clock_gating
 ...

****************************************
```

```
--------------------------------------------------------------------------------
                    Gated Register Report
--------------------------------------------------------------------------------
        Clock Gating Bank                |       Gated Register
--------------------------------------------------------------------------------
                                         |
clock_gate_2 (Level 1) (Pre-Existing)    | out_reg[4]
                                         | out_reg[5]
                                         | out_reg[6]
                                         | out_reg[7]
--------------------------------------------------------------------------------
 Total                                              4
--------------------------------------------------------------------------------
 Sum Total                          1 |              4
--------------------------------------------------------------------------------


                Clock Gating Summary
        ------------------------------------------------------------------
        |   Number of Clock gating elements       |          2      |
        |                                 |               |
        |   Number of Tool-Inserted Clock gates    |       1  (50.00%) |
        |                                 |               |
        |   Number of Pre-Existing Clock gates     |       1  (50.00%) |
        |                                 |               |
        |   Number of Gated registers              |       8  (80.00%) |
        |                                 |               |
        |   Number of Tool-Inserted Gated registers |       4  (40.00%) |
        |                                 |               |
        |   Number of Pre-Existing Gated registers  |       4  (40.00%) |
        |                                 |               |
        |   Number of Ungated registers            |       2  (20.00%) |
        |                                 |               |
        |   Total number of registers             |         10      |
        |                                 |               |
        |   Max number of Clock Gate Levels        |          1      |
        |                                 |               |
        |   Number of Multi Level Clock Gates      |          0      |
        ------------------------------------------------------------------
```

This example shows the report generated with the **-gating_elements** option in combination with the **-origin {tool_inserted}** option.

```
prompt> report_clock_gating -gating_elements -origin {tool_inserted}

****************************************
Report : clock_gating
 ...
\
****************************************


--------------------------------------------------------------------------------
                    Clock Gating Cell Report
--------------------------------------------------------------------------------

 Clock Gating Bank : clock_gate_out_reg_1 (Level 1) (TLATNTSCAX2MTL) (Tool-Inserted)
```

```
    INPUTS :
    [CLK  ] clock_gate_out_reg_1/CK = clk
    [EN   ] clock_gate_out_reg_1/E = en[0]
    [TE   ] clock_gate_out_reg_1/SE = *Logic0*

    OUTPUTS :
    [ENCLK] clock_gate_out_reg_1/ECK = gated_clk1
--------------------------------------------------------------------------------


                    Clock Gating Summary
  ------------------------------------------------------------------
  |   Number of Clock gating elements        |          2      |
  |                                          |                 |
  |   Number of Tool-Inserted Clock gates    |       1  (50.00%) |
  |                                          |                 |
  |   Number of Pre-Existing Clock gates     |       1  (50.00%) |
  |                                          |                 |
  |   Number of Gated registers              |       8  (80.00%) |
  |                                          |                 |
  |   Number of Tool-Inserted Gated registers |       4  (40.00%) |
  |                                          |                 |
  |   Number of Pre-Existing Gated registers  |       4  (40.00%) |
  |                                          |                 |
  |   Number of Ungated registers            |       2  (20.00%) |
  |                                          |                 |
  |   Total number of registers              |         10      |
  |                                          |                 |
  |   Max number of Clock Gate Levels        |          1      |
  |                                          |                 |
  |   Number of Multi Level Clock Gates      |          0      |
  ------------------------------------------------------------------
```

This example shows the additional information that is displayed in the summary report when the design contains multibit registers.

```
prompt> report_clock_gating

****************************************
Report : clock_gating
 ...

****************************************


                          Clock Gating Summary
 ------------------------------------------------------------------------------
 |                                          |     Bitwidth     |
 ------------------------------------------------------------------------------
 | Number of Clock gating elements        |          3      |                |
 |                                        |                 |                |
 | Number of Tool-Inserted Clock gates    |       3 (100.00%) |                |
 |                                        |                 |                |
 | Number of Pre-Existing Clock gates     |       0   (0.00%) |                |
 |                                        |                 |                |
 | Number of Gated registers              |       9  (81.82%) |       17  (89.47%) |
 |                                        |                 |                |
 | Number of Tool-Inserted Gated registers |       9  (81.82%) |       17  (89.47%) |
```

```
|                                |              |              |
| Number of Pre-Existing Gated registers |       0   (0.00%) |       0   (0.00%) |
|                                |              |              |
| Number of Ungated registers          |       2  (18.18%) |       2  (10.53%) |
|                                |              |              |
| Total number of registers           |        11     |         19     |
|                                |              |              |
| Max number of Clock Gate Levels       |         1     |              |
|                                |              |              |
| Number of Multi Level Clock Gates     |         0     |              |
-------------------------------------------------------------------------------
```

```
                Clock Gating Multibit Composition
        +-----------------------------------------------------------+
        |                        |  Actual   | Single-bit |
        |                        |  Count    | Equivalent |
        +-----------------------------------------------------------+
        |                        |           |            |
        | Gated Registers              |           |            |
        |              1-bit           |        5  |         5  |
        |              2-bit           |        2  |         4  |
        |              4-bit           |        2  |         8  |
        |             Total            |        9  |        17  |
        |                        |           |            |
        | Tool-Inserted Gated Registers  |           |            |
        |              1-bit           |        5  |         5  |
        |              2-bit           |        2  |         4  |
        |              4-bit           |        2  |         8  |
        |             Total            |        9  |        17  |
        |                        |           |            |
        | Pre-Existing Gated Registers   |           |            |
        |              1-bit           |        0  |         0  |
        |              2-bit           |        0  |         0  |
        |              4-bit           |        0  |         0  |
        |             Total            |        0  |         0  |
        |                        |           |            |
        | Ungated Registers              |           |            |
        |              1-bit           |        2  |         2  |
        |              2-bit           |        0  |         0  |
        |              4-bit           |        0  |         0  |
        |             Total            |        2  |         2  |
        |                        |           |            |
        | Total Registers              |           |            |
        |              1-bit           |        7  |         7  |
        |              2-bit           |        2  |         4  |
        |              4-bit           |        2  |         8  |
        |             Total            |       11  |        19  |
        |                        |           |            |
        +-----------------------------------------------------------+
```

This example shows the additional information that is displayed in the summary report when the design contains self gates.

```
prompt> report_clock_gating
****************************************
Report : clock_gating
 ...
```

```
****************************************

                Clock Gating Summary
-------------------------------------------------------------------
| Number of Clock gating elements        |           2      |
|                                        |                  |
| Number of Tool-Inserted Clock gates    |       2 (100.00%) |
|                                        |                  |
| Number of Pre-Existing Clock gates     |       0   (0.00%) |
|                                        |                  |
| Number of Gated registers              |      16 (100.00%) |
|                                        |                  |
| Number of Tool-Inserted Gated registers |      16 (100.00%) |
|                                        |                  |
| Number of Pre-Existing Gated registers  |       0   (0.00%) |
|                                        |                  |
| Number of Ungated registers            |       0   (0.00%) |
|                                        |                  |
| Total number of registers              |          16      |
|                                        |                  |
| Max number of Clock Gate Levels        |           1      |
|                                        |                  |
| Number of Multi Level Clock Gates      |           0      |
-------------------------------------------------------------------


                Self Gating Summary
-----------------------------------------------------------------
|   Number of self-gating cells       |           1      |
|                                     |                  |
|   Number of self gated registers    |       8  (50.00%) |
|                                     |                  |
|   Number of registers not self-gated |       8  (50.00%) |
|                                     |                  |
|   Total number of registers         |          16      |
-----------------------------------------------------------------
```

SEE ALSO

compile_fusion(2)
set_clock_gate_style(2)

report_clock_gating_check

Displays clock gating check information for the specified pins.

SYNTAX

status **report_clock_gating_checks**
[-significant_digits *digits*]
[-nosplit]
[*object_list*]

Data Types

digits int
object_list list

ARGUMENTS

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default.

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. **-nosplit** prevents line splitting and facilitates writing scripts to extract information from the report output.

object_list

Specifies a list of cells, pins, clocks or design objects for report.

DESCRIPTION

The **report_clock_gating_checks** command generates two tables. The first table displays information about all the clock gating checks, including cell, enable pin, clock pin, level, clock (check w.r.t. a specific clock, or all clocks if none is given), attribute (inferred, or icg), and what user settings have affected its S/H/R/F values and check level. The attribute indicates where the clock gating check was originally defined. It can be either inferred, or defined by library cell.

If there are multiple settings that affect a clock gating check, it is an error condition. The conflicts should be resolved by the user.

The second table displays information about user settings that affect the clock gating checks displayed in the first table, including netlist object of clock it is set on, S/H/R/F check values, level, and file/line where the relevant **set_clock_gating_check** and **remove_clock_gating_check** commands are defined. If there are no effective user settings that affect clock gating checks, the second table will not display.

Multicorner-Multimode Support

This command works only on the current scenario.

EXAMPLES

The following example reports all clock gating check information.

```
prompt> report_clock_gating_checks
****************************************
Report : clock_gating_check
Design : cgbox
Scenario: default
Version:
Date   :
****************************************
Updating Clock Gating Locations
  Inferring 6 clock-gating checks.

Clock gating checks:
Cell        EnablePin ClockPin  High/Low  Clock        Attr   User Settings
-------------------------------------------------------------------------------
U4          E       CK       Low                 L     1
U1          A       B        High                I     2
U2          B       A        High                I     2
U3          A       B        High                I     3
U5          A       B        High                I     3
U9          S0       B       Low                  I     4
U9          S0       A       Low      clk1          I     5

Attr: I:auto inferred, L:library cell defined


User clock gating settings:
              Rise         Fall
ID  Object    Setup  Hold   Setup  Hold   High/Low    File/Line
-------------------------------------------------------------------------------
1   U4/CK     --     --     --     --     Low         clock_gating2.tcl, line 79
2   U1/A      4.00   3.00   3.00   3.00   --          clock_gating2.tcl, line 87
3   U8/Y      2.00   2.00   2.00   2.00   --          clock_gating2.tcl, line 71
4   U9/S0     --     --     --     --     Low         clock_gating2.tcl, line 51
5   clk1      1.00   1.00   1.00   1.00   --          clock_gating2.tcl, line 61

1
```

SEE ALSO

set_clock_gating_check(2)
remove_clock_gating_check(2)
shell.common.report_default_significant_digits(3)

report_clock_gating_checks

Displays clock gating check information for the specified pins.

SYNTAX

status **report_clock_gating_checks**
[-significant_digits *digits*]
[-nosplit]
[*object_list*]

Data Types

digits int
object_list list

ARGUMENTS

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default.

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. **-nosplit** prevents line splitting and facilitates writing scripts to extract information from the report output.

object_list

Specifies a list of cells, pins, clocks or design objects for report.

DESCRIPTION

The **report_clock_gating_checks** command generates two tables. The first table displays information about all the clock gating checks, including cell, enable pin, clock pin, level, clock (check w.r.t. a specific clock, or all clocks if none is given), attribute (inferred, or icg), and what user settings have affected its S/H/R/F values and check level. The attribute indicates where the clock gating check was originally defined. It can be either inferred, or defined by library cell.

If there are multiple settings that affect a clock gating check, it is an error condition. The conflicts should be resolved by the user.

The second table displays information about user settings that affect the clock gating checks displayed in the first table, including netlist object of clock it is set on, S/H/R/F check values, level, and file/line where the relevant **set_clock_gating_check** and **remove_clock_gating_check** commands are defined. If there are no effective user settings that affect clock gating checks, the second table will not display.

Multicorner-Multimode Support

This command works only on the current scenario.

EXAMPLES

The following example reports all clock gating check information.

```
prompt> report_clock_gating_checks
****************************************
Report : clock_gating_check
Design : cgbox
Scenario: default
Version:
Date   :
****************************************
Updating Clock Gating Locations
  Inferring 6 clock-gating checks.

Clock gating checks:
Cell        EnablePin ClockPin  High/Low  Clock        Attr   User Settings
-------------------------------------------------------------------------------
U4          E       CK       Low                 L     1
U1          A       B        High                I     2
U2          B       A        High                I     2
U3          A       B        High                I     3
U5          A       B        High                I     3
U9          S0       B       Low                  I     4
U9          S0       A       Low      clk1          I     5

Attr: I:auto inferred, L:library cell defined


User clock gating settings:
                Rise         Fall
ID  Object    Setup  Hold   Setup  Hold   High/Low    File/Line
-------------------------------------------------------------------------------
1   U4/CK     --     --     --     --     Low         clock_gating2.tcl, line 79
2   U1/A      4.00   3.00   3.00   3.00   --          clock_gating2.tcl, line 87
3   U8/Y      2.00   2.00   2.00   2.00   --          clock_gating2.tcl, line 71
4   U9/S0     --     --     --     --     Low         clock_gating2.tcl, line 51
5   clk1      1.00   1.00   1.00   1.00   --          clock_gating2.tcl, line 61

1
```

SEE ALSO

set_clock_gating_check(2)
remove_clock_gating_check(2)
shell.common.report_default_significant_digits(3)

report_clock_gating_enable_condition

Reports clock gate enable signals.

SYNTAX

status **report_clock_gating_enable_condition**
-objects
[-nosplit]
[-flat]

ARGUMENTS

-objects

List of cells to be reported.

-nosplit

Option to display enable conditions on a single line.

-flat

Reports the enable signals of each clock gate through hierarchical ports.

DESCRIPTION

Reports the enable conditions of all clock-gating cells that are connected to registers or to another clock gate. A Boolean equation is used to represent the enable conditions as a sum of products: '&' is the AND operation, '|' is the OR operation and, '!' is the NOT operation. When describing an enable function, you might use reference points (invariant points) that require a Boolean variable. These reference points represent the following:

- Output pins of the register (since Q and QN pins are two different variables)
- Ports
- Output pins of any sequential cell, black box, or unsupported cell
- Hierarchical boundaries

EXAMPLES

The following example shows a report generated with **report_clock_gating** command issued with -objects option

```
prompt> report_clock_gating_enable_condition -objects [get_clock_gates]

--------------------------------------------------------------------------------
                    Enable Conditions Report
--------------------------------------------------------------------------------
Clock Gating Bank               |       Gated Register
 Clock_gate_1                   |       Register[1]
                                |       Register[2]
                                |       Register[3]
Enable condition:
!a |
b & c

--------------------------------------------------------------------------------
```

SEE ALSO

report_clock_gating(2)
replace_clock_gates(2)
compile_fusion(2)
set_clock_gate_style(2)

report_clock_gating_objects

Reports the object inclusion options that have been explicitly set for gating with **set_clock_gating_objects**.

SYNTAX

status **report_clock_gating_objects**
[-objects *object_list*]

Data Types

object_list list

ARGUMENTS

-objects *object_list*

Reports all objects in *object_list*, even if they have not been explicitly set for gating with **set_clock_gating_objects**.

DESCRIPTION

This command reports the settings that were explicitly set with the **set_clock_gating_objects** command. The report displays a table with the following columns:

- The action, set with **set_clock_gating_objects** command, the default values or its inherited values when applicable. The possible values are Include, Exclude or Force.
- The enable source option, set with **set_clock_gating_objects** command, the default values or its inherited values when applicable. The possible values are Both, Prefer Enable Pin, Prefer Feedback Loop, Enable Pin Only and Feedback Loop Only and "-".
- The instance for which the setting values are associated.

EXAMPLES

The following examples will use a design with two hierarchies (mid1, mid2), below the top level design, named top, and one hierarchie (bot) below mid2. The four, top, mid1, mid2, and top, have four registers each one.

The following example uses the next settings:

```
prompt> set_clock_gating_objects -include reg_top1 -enable_source enable_pin_only
prompt> set_clock_gating_objects -force reg_top3 -enable_source feedback_loop_only
prompt> set_clock_gating_objects -exclude mid1/reg_mid0
prompt> set_clock_gating_objects -include mid1/reg_mid1 -enable_source prefer_feedback_loop
prompt> set_clock_gating_objects -force mid1/reg_mid2 -enable_source prefer_enable_pin
prompt> set_clock_gating_objects -exclude mid1/reg_mid3
prompt> set_clock_gating_objects -include mid2/reg_mid0 -enable_source prefer_feedback_loop
prompt> set_clock_gating_objects -force mid2/reg_mid1 -enable_source prefer_feedback_loop
prompt> set_clock_gating_objects -exclude mid2/reg_mid2
prompt> set_clock_gating_objects -include mid2/reg_mid3 -enable_source feedback_loop_only
prompt> set_clock_gating_objects -force mid2/bot/reg_mid0 -enable_source prefer_enable_pin
prompt> set_clock_gating_objects -exclude mid2/bot/reg_mid1
prompt> set_clock_gating_objects -force mid2/bot/reg_mid3 -enable_source prefer_feedback_loop
prompt> set_clock_gating_objects -include mid2/bot -enable_source prefer_feedback_loop
prompt> set_clock_gating_objects -force mid2 -enable_source prefer_enable_pin
prompt> set_clock_gating_objects -force top -enable_source prefer_feedback_loop
```

The following example shows the generated report with **report_clock_gating_objects** command. Since the mid1 hierarchy does not have any explicit options set, it is not shown in the report.

```
prompt> report_clock_gating_objects

****************************************
Report: Clock-Gating Objects Report
Design: top
Version: 3.5.0
Date: Wed Sep 11 11:17:34 2019

****************************************


        Action          |       Enable Source       |         Instances
------------------------------+---------------------------+-----------------------------
        Forced          |    Prefer Feedback Loop   |           top
        Forced          |    Prefer Feedback Loop   |           mid2
        Forced          |    Prefer Feedback Loop   |         mid2/bot
        Include         |      Enable Pin Only      |         reg_top1
        Forced          |     Feedback Loop Only    |         reg_top3
        Exclude         |           Both            |       mid1/reg_mid0
        Include         |    Prefer Feedback Loop   |       mid1/reg_mid1
        Forced          |     Prefer Enable Pin     |       mid1/reg_mid2
        Exclude         |           Both            |       mid1/reg_mid3
        Include         |    Prefer Feedback Loop   |       mid2/reg_mid0
        Forced          |    Prefer Feedback Loop   |       mid2/reg_mid1
        Exclude         |           Both            |       mid2/reg_mid2
        Include         |     Feedback Loop Only    |       mid2/reg_mid3
        Forced          |     Prefer Enable Pin     |     mid2/bot/reg_mid0
        Exclude         |           Both            |     mid2/bot/reg_mid1
        Forced          |    Prefer Feedback Loop   |     mid2/bot/reg_mid3
```

SEE ALSO

set_clock_gating_objects(2)

report_clock_jitter

Reports clock jitter information previously set by the **set_clock_jitter** command.

SYNTAX

status **report_clock_jitter**
[-clock *clock_list*]

Data Types

clock_list list

ARGUMENTS

-clock *clock_list*

Specifies a list of clocks to report. If you do not use this option, the command reports the clock jitter for all the clocks whose master (or itself) has cycle-to-cycle jitter or duty-cycle jitter.

DESCRIPTION

This command reports the cycle-to-cycle jitter and the duty-cycle jitter of the clock. If the clock is a generated clock, it will report the cycle-to-cycle and the duty-cycle jitter of its primary master clock. For example, if the generated clock G2 is created from the generated clock G1 and the generated clock G1 is created from the primary master clock M, the cycle-to-cycle jitter and the duty-cycle jitter reported for G2 is in fact the cycle-to-cycle jitter and the duty-cycle jitter of its primary master clock, M. The command reports the primary master clock of the generated clock as well. In case the clock is primary master clock, it will print the clock itself.

To set the clock jitter use the **set_clock_jitter** command.

EXAMPLES

The following example reports the cycle-to-cycle jitter of 0.5 and duty-cycle jitter of 0.7 specified on the primary master clock mclk, and its generated clock gclk.

```
prompt> set_clock_jitter -clock [get_clocks mclk] -cycle 0.5 -duty_cycle 0.7
prompt> report_clock_jitter
```

```
****************************************
Report : clock jitter
 ...
****************************************

Clock        Cycle Jitter   Duty Cycle Jitter   Master Clock with Jitter
--------------------------------------------------------------------------------
mclk          0.500         0.700              mclk
gclk          0.500         0.700              mclk
```

SEE ALSO

remove_clock_jitter(2)
set_clock_jitter(2)

report_clock_power

Reports power information related to clock tree

SYNTAX

string **report_clock_power**
[-clocks *clock_names*]
[-modes *mode_names*]
[-corners *corner_names*]
[-scenarios *scenario_names*]
[-type per_segment | per_subtree]
[-nosplit]
[-significant_digits *digits*]

Data Types

clock_names list
mode_names list
corner_names list
scenario_names list
digits integer

ARGUMENTS

-clocks *clock_names*

Limits the reporting scope to the specified clock or collection of clocks. Clocks can be specified as a Tcl list or as a collection of clocks created by the *get_clocks* command. If specified as a Tcl list, all clock names refer only to clocks in the current mode. *get_clocks* also returns clocks only in the current mode by default. Use *get_clocks* mode to specify clocks for a particular mode other than the current mode.

-modes *mode_names*

Specifies the modes to be used for reporting. If this option is not given, the command reports all active modes for all active scenario. This is mutually exclusive to the **-scenario** option. This option cannot be used together with the **-clocks** option to specify a clock name for a specific mode. Instead, the **-clocks** option filters the output of this command to only the specified clocks, and the **-mode** option filters the output of this command to only the specified modes. To specify a clock name for a particular mode other than the current mode, use **-clocks [get_clocks <clock_name> -mode <mode_name>]**.

-corners *corner_names*

Specifies the corner to report. By default, reports all corners for active scenarios. This is mutually exclusive to the **-scenarios** option.

-scenarios *scenario_names*

Specifies the scenario to report. If the **-scenario** option is used, then **-mode** and **-corner** cannot be used. If none of **-modes**, **-corners**, or **-scenarios** options are specified, then the report is generated for all active scenarios.

-nosplit

By default, tabular reports are formatted to not exceed 80 columns in width. If certain fields exceed their fixed column width, then printing can automatically jump to the next line to maintain the formatting. This is done to improve readability of the report. If **-nosplit** is specified, then one row of data in a tabular report will always be printed on one row. The columns will no longer align, which affects readability of the report. The **-nosplit** option is useful to simplify parsing or post-processing of a report output file. If the report is not intended to be post-processed, then do not use the nosplit option, in order to get better report readability.

-significant_digits *digits*

Specifies the number of significant digits to show for reporting of any floating point number field. By default, each field has a particular number of significant digits, intended to communicate the right level of information to the user. This option overrides those default settings.

-type per_segment | per_subtree

Specifies the type of report to generate. Possible values of the -type argument are:

per_segment - A clock tree segment is defined as the clock buffer tree from a ICG to next ICG in its fanout. This option reports the power numbers per clock tree segments for each clock. Number of repeaters is the number of repeaters (both CTS added and pre-existing) in the clock tree segment. Number of sinks is the number of sinks driven by clock tree segment. Number of non-sink loads is number of non-sink loads driven by this clock tree segment. Total wire cap is sum of wire cap of nets in the clock tree segment. Total pin cap is sum of pin cap of all pins in the clock tree segment. Leakage power is sum of leakage power consumed by cells in this clock tree segment, including the driver, but excluding the segment loads. Net switching power is sum of net switching power of nets in the clock tree segment. Internal power is sum of internal power of pins in the clock tree segment (includes driver pin, and loads pins). The power numbers of macro are not included in the report as of now. Toggle rate of the driver pin will also be reported in a column.

per_subtree - A clock sub-tree is defined as the clock tree from a ICG all the way to sinks. This option reports the power numbers per clock subtree for each clock. Definitions of the column are same as that of per_segment.

DESCRIPTION

The **report_clock_power** command handles reporting of power numbers per clock tree segments and per clock subtree. The clock tree will be divided into segments or into subtree by pre-existing gates such as ICGs and the power numbers will be reported for the segment or subtree. A clock tree segment is defined as the clock buffer tree from a ICG to next ICG in its fanout, while a clock sub-tree is the clock tree from a ICG all the way to sinks. So there will be two kind of reports, one splitting the clock tree into clock segments and other into clock subtree. The user will be switch to per clock subtree reporting using a command option.numbers per clock tree segments and per clock subtree. The clock tree will be divided into segments or into subtree by pre-existing gates such as ICGs and the power numbers will be reported for the segment or subtree. A clock tree segment is defined as the clock buffer tree from a ICG to next ICG in its fanout, while a clock sub-tree is the clock tree from a ICG all the way to sinks. So there will be two kind of reports, one splitting the clock tree into clock segments and other into clock subtree. The user will be switch to per clock subtree reporting using a command option.

EXAMPLES

This example generates a summary report only for clock clk in mode modeA per segment.

```
prompt> report_clock_power -clocks [get_clocks clk -mode modeA] per_segment
```

SEE ALSO

report_clock_qor(2)
report_power(2)

report_clock_qor

Reports quality-of-results (QoR) information related to clock tree synthesis, including latency, skew, DRCs, area, power, multicorner robustness, and various histograms.

SYNTAX

string **report_clock_qor**
[-clocks *clock_names*]
[-skew_group *skew_group_names*]
[-from *pin_names*]
[-to *pin_names*]
[-through *pin_names*]
[-modes *mode_names*]
[-corners *corner_names*]
[-scenarios *scenario_names*]
[-type *rpt_type*]
[-csv summary | details]
[-output \ffilename]
[-histogram_type *hist_type*]
[-histogram_bins *number*]
[-histogram_min *number*]
[-histogram_max *number*]
[-show_paths]
[-show_verbose_paths]
[-trace_beyond_exception true | false | only]
[-largest *number*]
[-smallest *number*]
[-all]
[-robustness_corner *rcorner_name*]
[-per_clock_root]
[-nosplit]
[-significant_digits *digits*]

Data Types

clock_names list
skew_group_names collection
pin_names list
mode_names list
corner_names list
scenario_names list
rpt_type string
filename string
hist_type string
number integer
rcorner_name integer
digits integer

ARGUMENTS

-clocks *clock_names*

Limits the reporting scope to the specified clock or collection of clocks. Clocks can be specified as a Tcl list or as a collection of clocks created by the *get_clocks* command. If specified as a Tcl list, all clock names refer only to clocks in the current mode. *get_clocks* also returns clocks only in the current mode by default. Use *get_clocks* mode to specify clocks for a particular mode other than the current mode.

If a master clock is specified, then reporting is restricted to that clock and skew groups in that clock, no generated clocks are reported. If a generated clock is specified, then only that clock will be reported. No information about skew groups will be reported for the generated clocks, since skew groups belong to the master clock object. **-clocks** is mutually exclusive with the **-mode** and **-scenario** options.

-skew_group *skew_group_names*

Limits the reporting scope to the specified skew_group or collection of skew_groups. skew_groups can be specified as a Tcl list or as a collection of skew_groups created by the *get_clock_skew_groups* command. If specified as a Tcl list, all skew_group names refer only to skew_groups in the current mode. *get_clock_skew_groups* also returns skew_groups only in the current mode by default.

-skew_group option can be used in so many ways to get different results. For example, *-per_clock_root*, *-clocks*, *-corners*, options can be used with skew_group to again restrict the amount of result we get from *report_clock_qor*. **-skew_group** is mutually exclusive with the **-mode** and **-scenario** options.

-from *pin_names*

Limits the reporting scope to the fanout of the specified set of clock pins. This option can be used together with the **-to** and **-through** options. For **-type latency**, the latency of the sink will be reported as the latency calculated starting from the from-pin to the sink. If **-from** option is specified, all ideal clocks will ignored and will not be reported. It can accept hierarchical pins as valid inputs when **-type latency**/**structure**/**summary or -histogram_type latency** option is specified.

-to *pin_names*

Limits the reporting scope to the fanin to the specified set of clock pins. This option can be used together with the **-from** and **-through** options. It can accept hierarchical pins as valid inputs when **-type latency**/**structure**/**summary or -histogram_type latency** option is specified. In the case of **-type latency**/**robustness** and **-histogram_type latency**/**robustness**, the pins specified with this option should be endpoints and the non-endpoint pins specified will be ignored.

-through *pin_names*

Limits the reporting scope to only clock paths passing through the specified pin. This option can be used together with the **-from** and **-to** options. This option is not supported with **-type robustness**/**local_skew**/**balance_groups** and **-histogram_type robustness**/**local_skew**. It can accept hierarchical pins as valid inputs when **-type structure**/**latency**/**summary** and **-histogram_type latency** option is specified.

-modes *mode_names*

Specifies the modes to be used for reporting. If this option is not given, the command reports all active modes for all active scenario. This is mutually exclusive to the **-scenario** option. This option cannot be used together with the **-clocks** option to specify a clock name for a specific mode. Instead, the **-clocks** option filters the output of this command to only the specified clocks, and the **-mode** option filters the output of this command to only the specified modes. To specify a clock name for a particular mode other than the current mode, use **-clocks [get_clocks <clock_name> -mode <mode_name>]**.

-corners *corner_names*

Specifies the corner to report. By default, reports all corners for active scenarios. If more than one corner is specified using corner option, then multiple tables are printed (one each for every specified corners). This is mutually exclusive to the **-scenarios** option.

-scenarios ***scenario_names***

Specifies the scenario to report. If the **-scenario** option is used, then **-mode** and **-corner** cannot be used. If none of **-modes**, **-corners**, or **-scenarios** options are specified, then the report is generated for all active scenarios.

-type ***rpt_type***

Specifies the type of report to generate. Valid values are

- **summary**
  This is the default output if no type option is specified. It prints a summary table organized by clocks and skew groups, and includes basic summary information like area, buffer count, global skew, maximum latency, and logical DRCs. Use other reporting types to get more details about these metrics. For example, **-type summary** will show global skew, but **-type latency** must be used to get other skew information like boundary skew or local skew.

- **balance_groups**
  Reports a summary table of balance_groups in the design. There is no detailed table for this report. This reporting can be restricted to specific modes/corners/scenarios by giving options -corner/-mode/-scenarios. **balance_groups** can have master clocks or skew_groups as it's elements. -type balance_groups reports shows all information regarding balance_groups and it's elements- like target/scaled target offset, actual offset, max_latency, global skew etc.

- **latency**
  Reports a summary table of latency and skew information on all clocks and skew groups, including global / local / boundary skew, minimum / maximum / median latency, and latency and skew targets. The detailed section of the report shows sinks or balance points with their corresponding latencies. In order to see the latency to a hierarchical balance pin, user may specify hierarchical pins as input to **-to** option with this reporting.

- **drc_violators**
  Reports a summary table of transition and capacitance limit violations on the clock network, including total DRC count, cumulative DRC slack, and the DRC limits. The detailed section of the report shows clock pins with their corresponding DRC slacks.

- **robustness**
  Reports information about multicorner robustness of the CTS implementation. Robustness is a measure of how the latency to each clock sink scales between a measured corner and a reference corner. Every corner pair has a scaling factor, which is simply the ratio of the average latency in the measured corner over the average latency in the reference corner. The robustness value for an individual sink is the ratio of its latency in the measured corner over the latency in a reference corner, normalized against the scaling factor for those corners.

  If a clock sink has a robustness value of 1, that means it exhibits typical scaling between the measured and reference corner. A robustness value greater than one indicates a sink has higher than average latency in the measured corner compared to the reference corner. Likewise, a robustness value less than one indicates a sink with less than average latency in the measured corner compared to the reference corner. The largest and smallest robustness value sinks have the worst multicorner robustness, and could cause timing problems in the measured corner. If all clock sinks for a clock or skew group have a similar robustness value, then the clock tree is said to be multicorner robust, and the skew can be maintained across those corners.

  You must specify a robustness corner with the **-robustness_corner** option for this reporting type. The robustness corner represents the reference corner for measuring robustness values. Typically, the reference corner would be the corner that was used to synthesize the clock tree, or some dominant corner if multicorner CTS was used. The measured corners are determined based on the values specified by the corner or scenario arguments. When the measured corner and the reference corner are the same, the robustness value is meaningless and no information will be printed for that measured corner. If more than one corner is specified for the measured corner, then a separate table is printed for each measured corner. The output consists of a summary table for each clock and skew group, with information about the average latency, the scaling factor between the measured and reference corners, and then robustness values representing the extremes for that clock or skew group (minimum robustness value, maximum robustness value, and range). After that, a detailed table is

printed showing individual sinks with their corresponding robustness values.

- **area**
  Reports information about area and cell counts in the clock trees. The command first prints a summary table with a row for each clock and skew group. This table might include duplicate counting of area, for example, when multiple clocks share some of the same clock tree fanout. This section also includes a total area row at the end which provides overall area information for the reported scenario, and does not double-count any cells in the calculation.

  The command reports per-lib_cell area reporting in a separate section. This section analyzes exactly how many of each lib_cell are instantiated in the clock tree, and also organizes the cells by type (buffers, ICGs/muxes, macros vs standard cells). The **-smallest**, **-largest**, **-all**, and **-show_paths** options are incompatible with the area report.

- **power**
  Reports information about power in the clock trees. It is largely modeled on the area report, with some slight changes. There would be three sections to the power report: a per-scenario power table, per-lib_cell power tables for each reported scenario, and a per-hierarchy power table for each reported scenario. A report will be printed for each scenario whose setup/hold and leakage/dynamic power is enabled.

- **structure**
  The structure report is a very detailed representation of the clock tree structure. The entire fanout of the clock tree is represented in this report, with indentation and numbering indicating different levels of the clock tree. Clock balance points and ignore points and also represented in this report. It can accept hierachical pin as input to **-to/-through** options.

  report_clock_qor -type structure will show the balance point values on the pins. On a balance point pin, there can be multiple balance point values. For each corner exist in the design, we can set different values using the command **set_clock_balance_points**. For particular corner, there can exist different balance point values corresponding to early-rise, early-fall, late-rise and late-fall. Out of these four values minimum and maximum are printed. sample: [BALANCE PIN] [Offset values C1: max min C2: max min] where, C1 and C2 are corner names.

  The output of this report can be very large. As a result, it can be useful for parsing or searching for very specific information, but is generally not useful to get a high-level overview of the clock structure. For that purpose, you should use the Abstract Clock Graph feature of the GUI, which gives similar information to this report, but visually instead of textually. The **-smallest**, **-largest**, **-all**, **-show_paths**, and **-show_verbose_paths** options are incompatible with the structure report.

- **local_skew**
  This option prints the setup local skew and hold local skew information. Setup local skew is calculated as "launch flop latency - capture flop latency" and Hold local skew is calculated as "capture flop latency - launch flop latency". Reporting is done based on the scenario configuration. When setup=true, hold=false, only setup local skew is reported, when setup=false, hold=true, only hold local skew is reported and when setup=true, hold=true, both setup and hold local skews are reported.

This option is mutually exclusive with the **-histogram_type** option. The **-clocks**, **-from**, **-to**, **-modes**, **-corners**, and **-scenarios** options can be used to limit the scope of what is reported. The **-nosplit** and **-significant_digits** options affect all of the report output types.

Summary, latency, drc_violators and robustness reporting follows a similar format for the output. First, a summary table is shown with one row of output for each clock or skew group, with relevant QoR data printed. After the summary table, a detailed output section shows information about the worst clock pins, sinks, or balance points related to that metric. The number of items printed in this detailed section can be controlled using the **-largest**, **-smallest**, or **-all** options. The **-show_paths** option can be used to show a third category of output, which is a clock path report for each object reported in the detailed section.

-csv summary | **detail**

This option can be used to save the reports in CSV format. *-output* option should be specified with *-csv* to save the report as a file. This file will be saved in the current working directory. *-histogram_type* is not supported with *-csv* option.

Possible options for *-csv* are:

summary - to print the summary section of the report in CSV format. *details* - to print the details section of the report in CSV

format.

-output \ffilename

Using this option, user can specify the output filename to which the csv report will be dumped. This option can only be used with *-csv* switch. In case, tool can't create the file specified by the user, an error message will be displayed. It is recommended to save the output file with extension '.csv'.

-histogram_type *hist_type*

Specifies the type of histogram report to generate. This option is mutually exclusive with the **-type** option. The **-clocks**, **-from**, **-to**, **-through**, **-modes**, **-corners**, and **-scenarios** options can be used to limit the scope of what is reported. The **-significant_digits** option affects all of the report output types.

There are three additional options that can only be used with the **-histogram_type** option, but not the **-type** option. **-histogram_bins**, **-histogram_min**, and **-histogram_max** control the appearance of the output histograms. Generally, one histogram is printed for each clock or skew group. The output format for the histogram reports shows the histogram itself. The **-smallest**, **-largest**, and all options can be used to control the number of rows of detailed output per histogram. Detailed tables are printed only for the **-histogram_type** latency and robustness.

The possible options for *hist_type* are:

latency - Reports a histogram of clock latencies. The histogram includes one entry for every active clock edge at every clock sink or balance point. Latency can differ depending on whether a clock path to a sink is treated as the launch clock or capture clock. So, generating a latency histogram report actually reports two histograms per clock or skew group, one for launch clock latencies and the other for capture clock latencies.

transition - Reports a histogram of transition values in the clock network. The histogram includes one entry for every net load in the clock network, which could either be a cell input pin or an output port. Transition times for drivers of clock nets are not included in this histogram. Transition times at net sinks will always be greater than or equal to the transition time at the driver.

capacitance - Reports a histogram of total net capacitance values in the clock network. The histogram includes one entry for every clock net.

robustness - Reports a histogram of multicorner skew robustness values. The histogram includes one entry for every active clock edge at every clock sink or balance point. See the description of the **-type** robustness reporting type for more details about how robustness values are calculated. The **-robustness_corner** option must be used with this histogram type to specify a reference corner for calculating multicorner robustness values.

wire_delay_fraction - Reports a histogram of the wire delay fractions in the clock tree. The histogram includes one entry for every active clock edge at every clock sink or balance point. Wire delay fraction is defined as the ratio of the total wire delay over the total overall delay for a given path through the clock tree. Wire delay fraction is a useful metric for understanding the multicorner skew robustness of the clock tree. Typically, clock paths with similar wire delay fractions will have similar latency scaling between corners.

wire_delay - Reports a histogram of the wire delays in the clock tree.

cell_delay - Reports a histogram of the cell delays in the clock tree.

level - Reports a histogram of levels in the clock tree. The histogram includes one entry for every active clock edge at every clock sink or balance point. In a standard CTS flow, the clocks are not level-balanced. So, the level reporting is not a meaningful metric to analyze after CTS, except to look for extreme outliers in level count that could indicate an implementation problem. It might be more meaningful in custom CTS flows like a mesh or multisource. The level count can also be useful to analyze the clock tree before CTS, to understand how many gating or muxing levels are in each clock branch.

fanout - Reports a histogram of fanouts in the clock tree. The histogram includes one entry for every clock net. Fanout information is corner independent. Unlike most **report_clock_qor** reports, separate sets of histograms for each corner are not printed. If you specify both the **-corners** and **-histogram_type fanout** options, the command restricts which scenarios are considered for reporting.

wirelength - Reports a histogram of wirelengths in the clock tree. The histogram includes one entry for every clock net. The wirelength reported is the total wirelength of the net. Wirelength is calculated based on the current routing state of net (virtual, global, or detail routed).

Wirelength information is corner independent. Unlike most **report_clock_qor** reports, separate tables for each corner will not be printed. If you specify both the **-corners** and **-histogram_type wirelength** options, the command restricts which scenarios are considered for reporting.

wire_arc_length - Reports a histogram of wire arc lengths in the clock tree. The histogram includes one entry for every fanout of every clock net. The wire arc length is the wirelength just from the net driver to a particular net sink. Wire arc length is calculated based on the current routing state of net (virtual, global, or detail routed).

Wire arc length information is corner independent. Unlike most **report_clock_qor** reports, separate tables for each corner are not printed. If you specify both the **-corners** and **-histogram_type wire_arc_length** options, the command restricts which scenarios are considered for reporting.

local_skew - This report prints out a histogram of setup local skew and hold local skew distribution. Each entry in the histogram represent one launch and capture flop pair.

-histogram_bins ***number***

Divides the dataset into bins for reporting. This option specifies the number of bins in the histogram to report. The **-histogram_bins**, **-histogram_min** and **-histogram_max** options completely specify the number of bins and the range of values in each bin. The default value for **-histogram_bins** is 10, or the number of datapoints being reported, whichever is less. Although, if **-histogram_min** and/or **-histogram_max** is specified with **-histogram_bins**, we cannot always make bins accordingly. For example- **-histogram_bins** is 10, **-histogram_min** is 4 and **-histogram_max** is 6. This happens in the case of histograms are made with integer data points (for example, -histogram_type level).

-histogram_min ***number***

Specifies the smallest value to be reported in the histogram. Any data below this value is not added to a histogram bin. A total count of datapoints below the **-histogram_min** value is reported. By default, the **-histogram_min** value is equal to the smallest datapoint being reported.

-histogram_max ***number***

Specifies the largest value to be reported in the histogram. Any data above this value is not added to a histogram bin. A total count of datapoints above the **-histogram_max** value is reported. By default, the **-histogram_max** value is equal to the largest datapoint being reported.

-show_paths

Adds an additional section to the report that lists the clock path through the objects listed in the detailed section. This option can be used with the summary, latency, drc_violators, and robustness report types. Each of these reports shows a summary section, followed by a detailed section that reports the value of the metric on individual objects. An optional third reporting section is added by specifying this option. In this section, for every object reported in the detailed section, a clock path is reported through that object.

The path report is shown with the following commonly used columns for debugging clock paths: cell name, lib_cell name, fanout, capacitance, transition, increment, and arrival time.

The **-largest**, **-smallest**, and **-all** options apply to both the detailed section of these reports as well as the path section if the **-show_paths** option is specified. For example, the **report_clock_qor -type latency -largest 2 -show_paths** command shows the two largest latency sinks for each clock or skew group in the detailed section. Then, the paths associated with all those reported sinks will also be shown.

Unlike the **report_timing** report, the paths shown in **report_clock_qor** do not include the logical hierarchy pins, since they are seldom relevant to understanding the CTS implementation.

The path reports in **report_clock_qor** also have a 'Tags' column which annotates pins with useful information relevant to CTS.

The following annotations can be found in the 'Tags' column:

Scope Options
from: Marks from_pins
to : Marks to_pins
thru: Marks through_pins

Significant Clock Pins
clk : Marks master clock source pins
gclk: Marks generated clock source pins
sink: Marks default sink pins
icg : Marks ICG clock input pins
hier: Marks pins at the boundary or inside of physical hierarchy

Clock Exceptions
bal : Marks balance points
e_ign: Marks explicit ignore points
i_ign: Marks implicit ignore points

Clock Convergence
conv : Marks a pin where two or more clocks converge, such as a mux output
reconv: Marks a pin where a clock reconverges with itself

DRC Violations/Metrics
trans : Marks all pins with transition violations in the -type
drc_violators report. Marks the reported pin from the
-histogram_type transition report.

cap : Marks the driver pins of all nets with capacitance
violations in the -type drc_violators report. Marks the
driver pin of the reported net from the -histogram_type
capacitance report.

fanout: Marks the driver pins of all nets with fanout violations in
the -type drc_violators report. Marks the driver pin of the
reported net from the -histogram_type fanout report.

length: Marks the driver pins of all nets with net length violations
in the -type drc_violators report. Marks the driver pin of
the reported net from the -histogram_type wirelength
report. Marks the load pin of the reported wire arc from the
-histogram_type wire_arc_length report.

Dont_touch Constraints
dt_lib : set_dont_touch applied on lib_cell of the instance
dt_cell : set_dont_touch applied on the cell instance
dt_net : set_dont_touch applied on the net
dt_mv_cell: MV derived dont_touch on cell
dt_mv_net : MV derived dont_touch on net
dt_subtree: dont_touch_subtree derived dont_touch on cells and/or nets

If both **-show_paths** and **-show_verbose_paths** are specified, **-show_verbose_paths** takes precedence and more verbose paths are reported.

-show_verbose_paths

This option is similar to the **-show_paths** option, except it enables additional columns of output in the path report to help with

detailed debugging. If both **-show_paths** and **-show_verbose_paths** are specified, **-show_verbose_paths** takes precedence and more verbose paths are reported.

-show_verbose_paths generates path reports with the following columns in addition to the columns generated by **-show_paths**: location, non-default routing rules, and clock routing layer information.

The clock routing layer information is similar to the information provided by **get_attribute $net route_length**, in that it gives a per-layer breakdown of routing length for the net.

-trace_beyond_exception true | false | only

Using this option, how reporting is done beyond the exception pins can be manipulated for -type structure and drc_violators reports. Possible values for -trace_beyond_exception are follows: true : consider the clock tree beyond an exception pin. false: do not consider the clock tree beyond an exception pin. only : consider only the clock tree beyond the exception pin(including the exception pin).

-largest *number*

The latency, drc_violators, and robustness reports, along with robustness and latency histogram reports, show a detailed section where individual objects are reported. Each report type has a default behavior for the objects shown in this section. By using **-largest**, you can explicitly specify the number of objects to report in the detailed section, from the higher side of distribution of the metric. For example, **report_clock_qor -type latency -largest 5** reports the five largest latency sinks for each clock or skew group in the detailed section. **-largest** can be used together with **-smallest** to report objects from both sides of the distribution. If **-largest** and **-all** are specified together, **-all** takes precedence.

-smallest *number*

The latency, drc_violators, and robustness reports show a detailed section where individual objects are reported. Each report type has a default behavior for the objects shown in this section. By using smallest, you can explicitly specify the number of objects to report in the detailed section, from the lower side of the distribution of the metric. For example, **report_clock_qor -type latency -smallest 5** reports the five smallest latency sinks for each clock or skew group in the detailed section. **-smallest** can be used together with **-largest** to report objects from both sides of the distribution. If **-smallest** and **-all** are specified together, **-all** takes precedence.

-all

The latency, drc_violators, and robustness reports, show a detailed section where individual objects are reported. Each report type has a default behavior for the objects shown in this section. By using **-all**, every object will be shown in the detailed report section. For example, **report_clock_qor -type latency -all** reports all sinks or balance points for each clock or skew group in the detailed report section.

-robustness_corner *rcorner_name*

This option specifies a reference corner to measure against for the **-type robustness** and **-histogram_type robustness** reports. This is a mandatory option for both of those reports, as robustness reporting cannot occur without knowing what reference corner to measure against. See the description of the **-type robustness** report for more information about what the robustness number means and how it is calculated.

-per_clock_root

This option splits the reporting of a clock to per clock root, so that the clock trees associated with different root pins are reported separately. This option can be only used with **-type latency** option. The global skew/latency reported will be per clock root and hence will be calculated between sinks in the clock tree of a particular clock root. The longest/shortest path delay from root will be computed to get the global skew/latency for each root.

-nosplit

By default, tabular reports are formatted to not exceed 80 columns in width. If certain fields exceed their fixed column width, then printing can automatically jump to the next line to maintain the formatting. This is done to improve readability of the report. If **-nosplit** is specified, then one row of data in a tabular report will always be printed on one row. The columns will no longer align,

which affects readability of the report. The **-nosplit** option is useful to simplify parsing or post-processing of a report output file. If the report is not intended to be post-processed, then do not use the nosplit option, in order to get better report readability.

-significant_digits *digits*

Specifies the number of significant digits to show for reporting of any floating point number field. By default, each field has a particular number of significant digits, intended to communicate the right level of information to the user. This option overrides those default settings.

DESCRIPTION

=== Overview ===

The **report_clock_qor** command handles reporting of CTS-related QoR. This report has many subtypes, and can be used to get detailed information about area, DRCs, latency, skew, and multicorner robustness. Many histogram reports are also available to get details about metrics like latency, transitions, capacitance, and multicorner robustness. **report_clock_qor** reports the clock metrics for user analysis. No analysis is done by the command itself. See the **check_clock_trees** command which supports analysis regarding possible causes of bad CTS QoR. **report_clock_qor** does not report all clock settings, only a few that are relevant to the metric being reported, such as capacitance limits when reported capacitance DRC violations. For a complete report of clock settings that influence CTS behavior, see **report_clock_settings**.

=== CTS Timer View ===

report_clock_qor uses the CTS timer view, which is the same timer view that **synthesize_clock_trees** uses. In this view, only late mode timing constraints and derates are used for a given corner. This differs from the standard timing view, where for setup timing paths, late derates are applied to the launch clock path and early derates are applied to the capture clock path. The clock timing view applies the late derates only. This is particularly important for skew reporting. If skew is reported with early and late OCV derates applied, skew can appear very high due to the derates alone. For example, very high skew can be seen even at the same sink, due to different late and early derates applied to the path leading to the sink. This same-sink skew is unfixable by any CTS engine, and so it is preferable to run CTS and report CTS QoR with only late derates applied.

In the CTS timer view, timing path slack is not meaningful. It is only useful for analyzing clock paths themselves, rather than full timing paths. For clock related reporting that uses the standard timing view, see the command **report_clock_timing**.

=== Launch vs Capture Clocks ===

Even when the reporting is limited to just late mode constraints as described, it is still possible for the path to a sink for a given clock in a given corner to have differing launch vs capture latency values. This can happen for a variety of reasons, including clock reconvergence, float constraint exceptions applied with the **set_clock_balance_points** command, crosstalk, pin capacitance variation, and conditional delay arcs. These differing launch and capture clock path latencies contribute to the skew that must be minimized during clock optimization, and so need to be reported by **report_clock_qor** as well.

All latency and skew reporting fully considers the differing launch and capture latencies to each clock sink. In the detailed output for any report type that includes latency information, both the launch and capture latency to the sinks are given. Other metrics can also vary depending on whether analysis focuses on launch clocks or capture clocks. These metrics include transition values, pin capacitance, robustness, and wire delay fraction. These metrics are all reported for the launch clock only.

=== Clocks and Skew Groups ===

With the exception of the structure report, every reporting type and histogram type is organized by clocks and skew groups. Clocks can be master clocks created with the **create_clock** command, or generated clocks created with the **create_generated_clock** command. Every generated clock has a master, which creates a sort of hierarchy of clocks in the design.

Clocks define the topological structure of the clock trees synthesized by the CTS engine, but skew groups define the skew balancing behavior for those clocks, or for subsets of those clocks. Every time a **create_clock** statement is issued, a corresponding default skew group for that clock is created with the naming convention *default_<clock_name>*. By default, every sink of a master clock and

all of its generated clocks belong to that default skew group, meaning they will be skew-balanced together during CTS. The default skew group is owned by the master clock, but might contain sinks from other clocks generated from that master. Optionally, individual clock pins can be pulled into a separate user-defined skew group for balancing, by using the **create_clock_skew_group** command.

The clock and skew group information is reported by specifying the appropriate **report_clock_qor** output types. For example, the -type summary | latency | drc_violators | area | robustness reports all begin with a summary table of the same format, shown below:

```
Attributes
==========
M Master Clock
G Generated Clock
D Default Skew Group
U User Skew Group
* Generated Clock Balanced Separately

Clock / Skew Group   Attrs    Sinks <other_metrics>
-----------------------------------------------------
CLK1                 M          5000
  default_CLK1         D          4000
    GCLK1              G           500
    GCLK2              G          2500
      GCLK2_1          G           300
      GCLK2_2          G           200
    GCLK3              G          1000
  group1               U           600
  group2               U           400
CLK2                 M          6010
  GCLK4              G          5000
  GCLK5              G          1000
    GCLK5_1            G           400
```

The summary table shows the clock or skew group names, followed by a sink count, and then the relevant metrics, depending on the report type generated. histogram_type reports are similar, except a histogram is shown for each clock or skew group instead of a summary table.

In the above example, CLK2 is a simple case of a master clock with no user-defined skew groups. For brevity, the CLK2 line represents both the CLK2 clock defined by the **create_clock** command as well as the default skew group for that clock, called default_CLK2. Listed hierarchically under CLK2 is the generated clock structure for that master clock CLK2. CLK2 directly generates GCLK4 and GCLK5. GCLK5 then generates another generated clock GCLK5_1. The indentation indicates these relationships.

The more complex example is CLK1, which has user-defined skew groups. In this case, the sinks of CLK1 are divided among three skew groups: the default skew group default_CLK1, and the two user-defined skew groups group1 and group2. The CLK1 row represents the topological fanout of the master clock CLK1, and each indented skew group row represents the set of sinks balanced for that skew group. As expected, the sink count for CLK1 equals the sum of the sink counts for default_CLK1, group1, and group2 skew groups.

The generated clock structure for CLK1 is shown under the default_CLK1 skew group. This represents the generated clock topology and sinks belonging to the default skew group. Any generated clock sinks that are part of a user defined skew group are reported under that skew group, the same as skew grouped sinks of the master clock. So for example, there are 500 sinks in the fanout of GCLK1 belonging to the default_CLK1 skew group.

=== Cumulative Reporting ===

All metrics are reported cumulatively, meaning that the any metric reported for a given clock or skew group includes all of the generated clocks sourced from that clock. See the example below:

```
Clock/Skew Group  Attrs  Sinks  Latency  DRCs
```

```
-------------------------------------------------
CLK2            M    6010  2.012   104
 GCLK2_1        G    5000  1.293    54
 GCLK2_2        G    1000  0.884    41
  GGCLK2_2      G     400  0.431     9
```

CLK2 shows 6010 sinks. This includes the 5000 sinks of GCLK2_1 and the 1000 sinks of GCLK2_2. These numbers indicate that CLK2 has 10 direct sinks, but the other 6000 sinks are in the fanout of the two generated clocks. Likewise, GCLK2_2 has 1000 sinks, 400 in the fanout of its generated clock GGCLK2_2, and 600 in its direct fanout. All metrics are reported this way. Note that the latency for CLK2 is higher than that for GCLK2_2, which is larger than the latency for GGCLK2_2. This is because each latency value is calculated just from that clock root to its sinks. These numbers show that the source point of GCLK2_2 is (2.012 - 0.884) = 1.128ns deep in the CLK2 tree, and the source point of GGCLK2_2 is (0.884 - 0.431) = 0.453ns deep in the GCLK2_2 tree. Note that in the above example, the longest path is fanout for both GCLK2_2 and GGCLK2_2.

=== Balancing and Topological Metrics ===

The CTS metrics reported by the **report_clock_qor** command can be divided into two categories: balancing metrics and topological metrics. Balancing metrics are those directly related to single-corner or multicorner clock skew balancing: latency, skew, robustness, and wire delay fraction. Topological metrics are those related to the overall clock tree topology and structure. This includes drc violators (transition, capacitance), area and cell counts, transition times in the clock network, and capacitance on clock nets metrics.

Balancing metrics and topological metrics are handled differently by **report_clock_qor**. Balancing metrics are only reported for objects that are balanced, like skew groups. It does not make sense to report skew for objects that are not balanced, such as a master clock divided into several skew groups.

Topological metrics are only reported for clocks, not for skew groups. Skew groups are not guaranteed to have a clear and cohesive topology. There is no restriction on pins that can be added to a skew group, so it might consist of sinks from widely different branches of the clock tree.

See the example below, which is a truncated version of the **report_clock_qor -type summary** ***report.***

```
Clock /     Attrs  Sinks  Levels    Max  Global   DRC
Skew Group                       Latency   Skew  Count
-----------------------------------------------------------------
clk1        M     1450   10      1.548  0.132     3
clk2        M     2300   13       --    --        0
 default_clk2  D     2298   --      1.834  0.138    --
  genclk2     G     1560    7      0.840  0.110     2
 skew_group1   U        2   --      0.232  0.008    --
-----------------------------------------------------------------
All Clocks    --     3750   13       --    --        5
```

There are both balancing and topological metrics in this report. Latency and skew are balancing metrics, levels and DRC count are topological metrics. clk1 is a master clock with no user-defined skew groups, so that row represents both the topology of the clk1 fanout, as well as the default_clk1 skew group. So, both the balancing and topological metrics are reported on that row.

clk2 represents only the master clock fanout of clk2. For balancing, that clock has been divided into the default skew group default_clk2, and the user-defined skew group skew_group1. So, the clk2 row only reports topological metrics, and the default_clk2 and skew_group1 rows only report balancing metrics. genclk2 represents both the topology of the gen clk2 fanout, as well as the genclk2 sinks in the default skew group default_clk2. So, both the balancing and topological metrics are reported on that row.

=== Clock Tree Exceptions ===

Clock tree exceptions are constraints that provide exceptions to the default skew balancing behavior. There are two main categories of exceptions: balance points defined with the **set_clock_balance_points** command, and ignore points set with the **set_clock_balance_points -consider_for_balancing** command. Balance point exceptions force a pin to be balanced that is not normally a sink. Ignore point exceptions prevent a pin from being balanced that normally would have been.

By default, balancing metrics honor clock tree exceptions and topological metrics do not honor clock tree exceptions. Honoring the

exceptions means that the reporting will stop at any exceptions that exist in the middle of the clock tree. Not honoring exceptions means that the reporting will trace beyond exceptions to the original sink pins of the clock network.

It makes sense for balancing metrics to honor the exceptions, because the exceptions were applied in order to guide the balancing behavior during CTS. It makes sense for topological metrics to disregard the exceptions, because the parts of the clock tree beyond exceptions, even though they are not balanced, are still considered to be part of the clock network, and will be synthesized during CTS. Metrics like DRCs and area (and other topological metrics) should consider that part of the clock network during reporting.

You can override the default exception handling by setting the report_beyond_exceptions option to true or false. The default behavior is true, which works as described above.

=== How Clock Metrics Are Calculated ===

report_clock_qor reports on many different CTS related metrics. The method for calculating some of these metrics requires further explanation:

Sinks: Sink count is shown in many **report_clock_qor** sub-reports. This metric is taken to mean 'sink pin' count. In other words, if a cell has multiple clock inputs, it will count as a sink for each clock input. Similarly, some abstracts can have a single physical clock input pin that fans out to multiple check pins inside the abstract. Each of these check pins count as one sink. Also, sink count is calculated while honoring clock exceptions like balance points and ignore points. So, sink count is representative of what is actually balanced during CTS, not the physical leaf sink pins in the clock network.

For example, if a leaf-level register on the clock network has an ignore point constraint on it, it is not reported as a sink in most **report_clock_qor** reports. Likewise, an intermediate pin in the clock network might have a balance point constraint on it, in which case it will be considered a sink, and all downstream physical sinks will not be considered sinks for balancing or reporting.

But, for the area and power reports, sink count has a different meaning. Sink counts in those reports are based on sink instances rather than sink pins. So a cell with multiple inputs would count only one time as a sink in the area or power report.

Additionally, the area and power reports consider sinks to be the physical leaf-level sinks in the clock tree, regardless of any balance point or ignore point constraints on the design. For example, when the power report shows 'Standard Cell Sinks' and 'Macro Sinks', these are referring to the physical leaf-level sinks rather than skew-balancing sinks.

The area report in some cases shows both types. 'Sinks (balanced)' means the sink cells that are balanced by CTS while honoring balance point and ignore point constraints. 'Sinks (physical)' means the physical leaf-level sinks.

Area: The **-type summary** report includes three area related columns: Buffer Count, Buffer Area, and Total Area. Buffer Count and Buffer Area mean the count and area of buffers or inverters inserted by clock tree synthesis. This could be any part of the CTS solution: synthesis, optimization, balance group balancing, multisource synthesis, trunk planning, concurrent clock and data, and so on. If a buffer or inverter is preinstantiated in the netlist, or manually added to the clock tree, or inserted by datapath optimization, it does not count toward the Buffer Count and Buffer Area columns. The Total Area column includes every cell in the clock tree except for the sinks.

The **-type area** report includes much more area information. The per-clock and per-hierarchy sections of the report include the following columns:

- Sinks (balanced): This is the number of sinks and balance points considered for CTS balancing. This is the same as the 'Sinks' column in all of the other **report_clock_qor** types.
- Sinks (physical): This is the number of leaf sink pins in the clock network, disregarding balance point and ignore point constraints. This sink count is more relevant to the area calculation since area reporting normally ignores exceptions.
- Clock Cell Count (non-sink): Total count of all standard cells and macros in the clock network, excluding physical sinks.
- Clock Stdcell Area: The area of standard cells from 'Clock Cell Count (non-sink)'.
- Clock Macro Area: The area of macros from 'Clock Cell Count (non-sink)'.
- Clock Repeater Count: The total count of buffers or inverters in the clock network that were inserted by clock tree synthesis commands. Preexisting repeaters, or those inserted by datapath optimization commands, or those imported from another tool

are not included in this count.

- Clock Repeater Area: The area corresponding to 'Clock Repeater Count'.
- Sink Stdcell Area: The area of standard cells from the 'Sinks (physical)' category.
- Sink Macro Area: The area of macro cells from the 'Sinks (physical)' category.

Levels: Levels are counted from the clock root to the physical sinks of the clock tree, tracing beyond any intermediate balance point or ignore point exceptions. The master clock driver itself does not count as a level (it is level 0), so the number of levels is the number of logic stages traversed in the clock tree plus the sink level. Physical and logical hierarchy pins do not count as levels.

Latency and Skew: Latency is calculated based on the total delay from the clock root to the clock sink. If the clock root has a **set_clock_latency** source latency constraint applied, this value is not included in the latency reported by **report_clock_qor**. Latency can be adjusted for clock balance point offsets applied with the **set_clock_balance_points** command. For example, if the actual delay from the clock root to the sink is 2ns, and there is a +0.5ns offset applied on the sink using **set_clock_balance_points** (meaning that the sink should be tapped 0.5ns early in the clock tree), then the reported latency in **report_clock_qor** will be 2.5ns. By making this adjustment, these intentional balancing offsets will not appear as skew in the report, if CTS has implemented the offsets successfully.

Latency can be different depending on whether a clock sink is used to launch or capture a path. For the **-type latency** and **-type summary** reports, both launch and capture latency are considered for calculating the latency and skew values. The detailed section of the report shows both the launch and capture latency for each reported sink. In the **-histogram_type latency** report, separate histograms can be shown for each clock indicating launch and capture latency.

Robustness: The **-type robustness** and **-histogram_type robustness** reports use a robustness metric that is assigned for every sink. Robustness is a way of evaluating the multicorner robustness of the clock tree, which is important for maintaining low skew in all signoff corners.

Robustness is a measure of how the latency to each clock sink scales between a measured corner and a reference corner. For this reason, a **-robustness_corner** must always be specified when generating a robustness report, to act as the reference corner.

First, an unnormalized robustness value is calculated for every clock sink, which is the ratio of (latency in measured corner) / (latency in reference corner). Additionally, a scaling factor is determined for the corner pair, which is the ratio of (median latency of all sinks in measured corner) / (median latency of all sinks in reference corner). This scaling factor indicates the typical scaling that occurs between those two corners. For example, the scaling factor of holdCorner compared to setupCorner might be 0.5, indicating that the typical clock sink has half the latency in the holdCorner compared to setupCorner.

The robustness numbers for each sink are normalized against this scaling factor. As a result, if a clock sink has a robustness value of one, that means it exhibits typical scaling between the measured and reference corner. A robustness value greater than one indicates a sink has higher than average latency in the measured corner compared to the reference corner. Likewise, a robustness value less than one indicates a sink with less than average latency in the measured corner compared to the reference corner.

When doing multicorner skew analysis, it is useful to look at the sinks that are robustness outliers on either the high or low side. These are the sinks that exhibit atypical scaling between those corners, which can cause higher than expected skew. If all clock sinks for a clock or skew group have a similar robustness value, then the clock tree is said to be multicorner robust, and the skew can be maintained across those corners.

"All Clocks" Reporting: The summary tables generally have an 'All Clocks' row at the end. Likewise, histogram reports typically have an 'All Clocks' histogram reported after all the clock and skew group specific histograms.

"All Clocks" in most cases is self-explanatory. Consider the case of two clocks muxed together with an overlapping sub-tree. The 'All Clocks' row or histogram will consider the full clock trees of both clocks, and calculate metrics across all clocks without double-counting. For example, the area would be calculated in such a way that the common logic of the two clocks is not double-counted. So the total area for "All Clocks" would be less than the individual area of the two clocks summed together. So, "All Clocks" gives a simple overview of your entire clock structure for that particular reported corner.

"All Clocks" data is shown per-corner. So for example, all modes of a given corner would be considered for the "All Clocks" summary table row or histogram.

In a few cases, the meaning of "All Clocks" isn't clear. In other cases, no "All Clocks" metric is shown, because it is not very meaningful. This list describes the "All Clocks" metrics that might not be intuitive:

- DRCs / Transition / Capacitance: If a given pin or net has multiple transition, capacitance, or DRC values due to multiple clocks, then only the largest is selected for "All Clocks".
- Robustness Max: Takes the maximum robustness value from any one clock or skew group
- Robustness Min: Takes the minimum robustness value from any one clock or skew group
- Robustness Range: Shows the difference between Robustness Max and Robustness Min for the "All Clocks" row, rather than showing the largest range for any one skew group.
- Global/Local/Boundary Skew: The largest global/local/boundary skew of one clock or skew group. This does not calculate a new skew value based on the dataset of all sinks in all skew groups. It does not make sense to report a skew value across a dataset that was never skew balanced.
- Target Skew: Shows the global (non-clock-specific) target skew value set with the **set_clock_tree_options** command.
- Target Latency: Shows the global (non-clock-specific) target latency value set with the **set_clock_tree_options** command.

=== Using **report_clock_qor** Options ===

To use **report_clock_qor**, first select the type of report to generate. By default, a summary report is generated, which contains useful summary information about the area, latency, skew, and DRCs in the clock tree. Other reporting types can be generated with the type and histogram_type options.

After selecting the reporting type, consider the scope of the report. By default, all clocks and skew groups in all active CTS scenarios are reported. A CTS scenario is any scenario set true for setup, hold, max_transition, or max_capacitance fixing using the **set_scenario_status** command. Use **-clocks** to limit the reporting to only certain clocks. Use **-modes** and **-corners**, or **-scenarios** to limit reporting to only certain modes, corners, or scenarios. Use **-from**, **-to**, and **-through** to limit reporting only to clock paths from, to, or through certain pins.

If you generate a histogram report, the **-histogram_bins**, **-histogram_min**, and **-histogram_max** options can be used to control the characteristics of the histograms generated.

Many of the reporting types, such as the latency, drc_violators and robustness, have a summary section followed by a detailed section. The summary section shows either a summary table or a set of histograms for the reported metrics. These are partitioned by clock and skew group. The detailed section shows detailed information about the outlier objects for that metric. For example, **report_clock_qor -type drc_violators** would have a detailed section that shows the clock pins or nets that have the highest DRC slack. The number of objects printed in the detailed report section can be controlled with the **-smallest**, **-largest**, and **-all** options.

If the **-smallest**, **-largest**, or **-all** options are not used, every report type has a default behavior. The drc_violators report shows all DRC violations by default. Every other report that has a detailed section shows the five largest and five smallest datapoints for the metric, by default.

For the reports with a detailed section, the **-show_paths** option can be used. This option shows an additional section after the detailed section, with a full clock path report through each object in the detailed section. For example, **report_clock_qor -type latency -largest 1 -show_paths** would show the largest latency sink for each clock and skew group in the detailed section. It would also show the paths to each of those sinks.

For **-type robustness** and **-histogram_type robustness** reports, the **-robustness_corner** option must be specified. This option gives the reference corner against which the robustness value is calculated for each sink in the measured corner. See the description of the type robustness report for more information about how the robustness numbers are calculated and what they mean.

Multicorner-Multimode Support

EXAMPLES

This example generates a summary report only for clock clk in mode modeA.

```
prompt> report_clock_qor -clocks [get_clocks clk -mode modeA]
```

This example uses many scope limiting options to restrict what is returned by the report.

```
prompt> report_clock_qor -clocks clk -scenario scenarioA -through myICG/CK \
  -to sink1/CK
```

This example generates a clock robustness report in cornerA with cornerB used as the reference corner.

```
prompt> report_clock_qor -type robustness -corner cornerA -robustness_corner \
  cornerB
```

In the following example, the **report_clock_qor** command is limited to the scope of only those sinks clocked by clk, and only within the fanout cone of pin *c/Z*.

```
prompt> report_clock_qor -clocks clk -from c/Z -type latency
```

In the following example, the **report_clock_qor** command writes out the 5 largest and 5 smallest latency clock sinks.

```
prompt> report_clock_qor -clocks clk -type latency -largest 5 -smallest 5
```

In the following example, the **report_clock_qor** command shows all the DRC violators.

```
prompt> report_clock_qor -clocks clk -type drc_violators
```

In the following example, the **report_clock_qor** command reports largest and smallest latency sinks.

```
prompt> report_clock_qor -clocks clk -type latency
```

In the following example, the report_clock_qor command reports all latency sinks.

```
prompt> report_clock_qor -clocks clk -type latency -all
```

In the following example, the **report_clock_qor** command reports all latency sinks. The **-largest** and **-smallest** options are ignored.

```
prompt> report_clock_qor -clocks clk -type latency -all -largest 10 \
  -smallest 10
```

SEE ALSO

check_clock_trees(2)
create_clock(2)
create_clock_skew_group(2)
create_generated_clock(2)
report_clock_settings(2)
report_clock_timing(2)
report_timing(2)
set_clock_balance_points(2)
set_clock_tree_options(2)
set_scenario_status(2)
set_timing_derate(2)

synthesize_clock_trees(2)

report_clock_routing_rules

Reports clock routing rules used in clock tree synthesis.

SYNTAX

status **report_clock_routing_rules**

DESCRIPTION

This command reports routing rules used in clock tree synthesis. The clock routing rules are set by **set_clock_routing_rules** command.

EXAMPLES

The following example assigns a non-default rule named *dblespacing* to the sink level nets of clock clk1. Then it reports the routing rule to confirm the settings.

```
prompt> set_clock_routing_rules -clocks clk1 -net_type sink -rule dblespacing
prompt> report_clock_routing_rules


****************************************
 Report : clock routing rules
 Design : Decoder
 Date   : Thu Nov 29 12:28:00 2012
****************************************
  Clock: clk1 (mode default); Net Type: sink;     Rules: dblespacing
```

SEE ALSO

set_clock_routing_rules(2)
remove_clock_routing_rules(2)

report_clock_settings

Reports existing clock settings and constraints.

SYNTAX

status **report_clock_settings**
[-clocks *clock_list*]
[-type configurations | routing_rules | references | spacing_rules | all]
[-significant_digits *digits*]
[-nosplit]

Data Types

clock_list list
digits integer

ARGUMENTS

-clocks *clock_list*

Reports only the settings on the clocks specified in *clock_list*. By default, the command reports for all currently defined clocks in all active modes.

-type configurations | routing_rules | references | spacing_rules | all

Specifies which type of reporting to generate. There are five types specified by keywords **configurations**, **routing_rules**, **references**, **spacing_rules** and **all**. **configurations** refers to clock tree synthesis configurations, including max transition and max capacitance specified by **set_max_transition** and **set_max_capacitance**; **routing_rules** refers to clock tree routing rules specified by **set_clock_routing_rules**; **references** refers to available buffers and invertors in clock tree synthesis; **spacing_rules** refers to clock cell spacing rules specified by **set_clock_cell_spacing**; **all** refers to all the above types of settings. The default value of this option is **all**.

-significant_digits *digits*

Specifies the number of digits after the decimal point to be displayed for time values in the generated report. Allowed values are 0-13. The default is 3. Use this option if you want to override the default.

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. If this option is not used, most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_clock_settings** command reports existing clock settings and constraints. By default, the **report_clock_settings** command reports clock configurations, routing rules, and references.

Multicorner-Multimode Support

This command works only on the mode associated with the current scenario.

EXAMPLES

In the following example, the **report_clock_settings** command reports routing rules specified on a clock called clk1.

```
prompt> report_clock_settings -type routing_rules -clocks clk1
```

SEE ALSO

set_clock_cell_spacing(2)
set_clock_routing_rules(2)
set_max_capacitance(2)
set_max_transition(2)

report_clock_skew_groups

Reports the user-defined skew groups in the design.

SYNTAX

status **report_clock_skew_groups**
[*skew_group_list*]
[-nosplit]

Data Types

skew_group_list collection

ARGUMENTS

skew_group_list

List of skew group names/collections.

-nosplit

Indicates that lines are not to be split when column fields overflow. This is most useful for doing diffs on previous scripts, or for post-processing the script.

DESCRIPTION

The command reports the user-defined skew groups in the design. If you dont specify any skew groups then all of the ones in the current mode are reported.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example lists all skew groups that are on clock path clk1.

prompt> **report_clock_skew_groups [get_clock_skew_groups -of_objects clk1]**

```
****************************************
Report   : Skew group
Design   : Hier_design

****************************************


==============Skew Group Data================

 1. Skew Group Name : A
   Pin List : 10
      A/B/C/FF4/CLK
      A/B/C/FF3/CLK
   Clock List :
      clk1

 2. Skew Group Name : B
   Pin List : 10
      A/B/C/FF1/CLK
      A/B/C/FF2/CLK
   Clock List :
      clk1

1
```

SEE ALSO

create_clock_skew_group(2)
remove_clock_skew_groups(2)
get_clock_skew_groups(2)

report_clock_timing

Reports timing attributes of clock networks.

SYNTAX

string **report_clock_timing**
  -type *report_type*
  [-modes *mode_list*]
  [-corners *corner_list*]
  [-scenarios *scenario_list*]
  [-clock *clock_list*]
  [-from_clock *from_clock_list*]
  [-to_clock *to_clock_list*]
  [-to *to_list*]
  [-from *from_list*]
  [-through *through_list*]
  [-setup | -hold]
  [-launch | -capture]
  [-rise | -fall]
  [-nworst *worst_entries*]
  [-greater_than *lower_limit*]
  [-lesser_than *upper_limit*]
  [-slack_lesser_than *slack_upper_limit*]
  [-include_uncertainty_in_skew]
  [-verbose]
  [-nets]
  [-physical]
  [-show_clocks]
  [-derate]
  [-variation]
  [-clock_crossing]
  [-clock_synthesis_view]
  [-nosplit]
  [-significant_digits *digits*]

Data Types

report_type string
mode_list list
corner_list list
scenario_list list
clock_list list
from_clock_list list
to_clock_list list
from_list list
to_list list
through_list list

worst_entries int
lower_limit float
upper_limit float
slack_upper_limit float
digits int

ARGUMENTS

-type *report_type*

Specifies the type of report to be generated. Allowed values are as follows:

transition : Specify a transition time report.

latency : Specify a latency report. Note that only clock paths that reach right trigger edge at endpoint will be reported.

skew : Specify a skew report; you cannot use the *-launch*, *-capture*, *-rise*, *-fall*, and *-lesser_than* options if you specify a skew report, and you can use the *-include_uncertainty_in_skew* option only in a skew, interclock_skew, or summary report.

interclock_skew : Specify an interclock skew report; you cannot use the *-launch*, *-capture*, *-rise*, *-fall*, and *-lesser_than* options if you specify an interclock skew report, and you can use the *-include_uncertainty_in_skew* option only in a skew, interclock_skew or summary report.

summary : Specify a summary report, which shows the worst instances of transition time, latency and skew over the clock networks or subnetworks of interest. If you specify a summary report, you can use only the *-clock*, *-to_list*, *-from_list*, *-include_uncertainty_in_skew*, *-nosplit*, and *-significant_digits* options.

For nonsummary reports, report entries are ordered with respect to the specified attribute of interest (transition time, latency, or skew). Note that all skews reported are "local" skews. For an explanation of local skew, see the DESCRIPTION section.

-modes *mode_list*

Specifies the modes to be used for reporting. If this option is not specified, the command uses the current mode. For each mode, a subreport will print the worst value across all specified scenarios of that mode. It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners to be used for reporting. For each mode being reported, scenarios (if any) of that mode and the corners specified by the **-corners** option will be used for reporting. If this option is not given, the command uses every active scenario of the specified modes. It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios to be used for reporting. Each mode used by the specified scenarios is reported. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the report uses all scenarios of the current mode.

-clock *clock_list*

Specifies a list of clocks to be used in the report. A subreport is typically produced for every clock in the *clock_list*, unless you additionally specify the *to_list*, *from_list*, or both options that has no network intersection with a given clock. In that case, the report removes these clocks from the *clock_list* and issues a warning. By default, if you do not specify *clock_list*, all clocks in the design that have associated clock networks are used in the report. If a clock in the list does not belong to the mode being reported, the command instead tries to use a clock from that mode with the same name as the given clock.

-from_clock *from_clock_list*

Specifies a list of clock networks to be used as from-clocks in the current interclock skew report. This option can be used only in an interclock skew report. The report typically considers every clock in *from_clock_list*, unless you additionally specify the *from_list* option that has no network intersection with a given clock. In that case, the report drops these clocks from the *from_clock_list* and also issues a warning. By default, if you do not specify the *from_clock_list* option, all clocks in the design that have associated clock networks are used as from-clocks in the report. If a clock in the list does not belong to the mode being reported, the command will instead try to use a clock from that mode with the same name as the given clock.

-to_clock *to_clock_list*

Specifies a list of clock networks to be used as to-clocks in the current interclock skew report. This option can be used only in an interclock skew report. The report typically considers every clock in *to_clock_list*, unless you additionally specify a *to_list* that has no network intersection with a given clock. In that case, the report removes these clocks from the *to_clock_list* and also issues a warning. By default, if you do not specify the *to_clock_list* option, all clocks in the design that have associated clock networks are used as to-clocks in the report. If a clock in the list does not belong to the mode being reported, the command instead tries to use a clock from that mode with the same name as the given clock.

-to *to_list*

Specifies the list of sequential clock network pins or ports to consider as to-pins in the current report. If any named pin is not the clock pin of a sequential device, such as a sink pin, the report replaces that pin with all sequential clock pins in the transitive fanout of the named pin. If there are no sequential clock pins in the pin's transitive fanout, the pin is dropped from the list with a warning message.

For skew reports, the from-to skew sense is defined by the pins in the *from_list* and *to_list* options respectively; if the *to_list* option is not specified, by default all sink pins in the given clock networks are used. Thus, specifying the *to_list* option reduces the topological scope of the report.

For transition time and latency reports, the *from_list* and *to_list* options are concatenated and treated as a single list; if neither is specified, the *to_list* option is assumed to be populated with all sink pins in the given clock networks.

-from *from_list*

Specifies a list of sequential clock network pins or ports to consider as from-pins in the current report. If any named pin is not the clock pin of a sequential device, the report replaces that pin with all sequential clock pins in the transitive fanout of the named pin. If there are no sequential clock pins in the pin's transitive fanout, the pin is dropped from the list and a warning message is issued.

For skew reports, the from-to skew sense is defined by the pins in the *from_list* and *to_list* options respectively. If the *from_list* option is not specified, by default all sink pins in the given clock networks are used. Thus, specifying the *to_list* option reduces the topological scope of the report.

For transition time and latency reports, the *from_list* and *to_list* options are concatenated and treated as a single list. If neither is specified, the *to_list* option is assumed to be populated with all sink pins in the given clock networks.

-through *through_list*

Reports only clock paths that pass through the named pins, ports, cells, or nets. This option may only be used with **-type latency** and is exclusive with *-nworst* and *-clock_synthesis_view* options. If you specify only one **-through** option, the tool reports only the paths that travel through one or more of the objects in the list. You can specify **-through** multiple times with one **report_clock_timing** command. This option triggers verbose report automatically. The **-through** option may not specify an object on the network between master clock sources and their generated clock sources.

-setup

Specifies that only the setup data path is to be used in the report, and that the skews, latencies or transition times reported must correspond to those used by the **report_timing** command in the verification of setup constraints. The *-setup* and *-hold* options are mutually exclusive. If neither option is specified, the *-setup* option is assumed. You cannot use this option in summary reports.

-hold

Specifies that only the hold data path is to be used in the report, and that the skews, latencies, or transition times reported must correspond to those used by the **report_timing** command in the verification of hold constraints. The *-setup* and *-hold* options are mutually exclusive. If neither option is specified, the *-setup* option is assumed. You cannot use this option in summary reports.

-launch

Specifies that only pins launching data are to be used in the report, and that latencies or transition times reported must correspond to those used by the **report_timing** command for sequential device clock pins that are launching data. In skew reports, the role is implicit from the from-to sense indicated by the *from_list* and *to_list* options. In all other reports, the *-launch* and *-capture* options are mutually exclusive. If neither option is specified, the *-launch* option is assumed. You cannot use this option in summary or skew reports.

-capture

Specifies that only pins capturing data are to be used in the report, and that the latencies or transition times reported must correspond to those used by the **report_timing** command for sequential device clock pins that are capturing data.

In skew reports, the role is implicit from the from-to sense indicated by the *from_list* and *to_list* options. In all other reports, the *-launch* and *-capture* options are mutually exclusive. If neither option is specified, the *-launch* option is assumed. You cannot use this option in summary or skew reports.

-rise

Specifies that the active transition is a rising edge for sequential device clock pins in the current report. The *-rise* and *-fall* options are mutually exclusive; if neither option is specified, the active transition at a latch or flip-flop is deduced from the launch or capture role and the behavior of the sequential device itself. This option enables you to answer "what if" questions regarding latency and transition time at sequential device clock pins. You cannot use this option in summary or skew reports.

-fall

Specifies that the active transition is a falling edge for sequential device clock pins in the current report. *-rise* and *-fall* are mutually exclusive; if neither is specified, the active transition at a latch or flip-flop is deduced from the launch or capture role and the behavior of the sequential device itself. This option enables you to answer "what if" questions regarding latency and transition time at sequential device clock pins. You cannot use this option in summary or skew reports.

-nworst *worst_entries*

Specifies the number of worst report entries to be reported per clock domain; the default is 1. Entries are sorted with respect to the attribute of interest, such as transition time, latency or skew and so on.

The worst entries are those most likely to cause a violation. For example, if you request a latency report using *-setup* and *-capture*, the smallest *worst_entries* are listed, sorted in ascending order, because small capture latencies for setup paths are more likely to lead to a violation than large capture latencies. By contrast, for skew reports, the worst entries always correspond to the largest skews over the specified domain. You cannot use this option in summary reports.

-greater_than *lower_limit*

Specifies that only those entries whose attribute value (latency, transition time or skew) is greater (more positive) than *lower_limit* are to be shown. You cannot use this option in summary reports.

-lesser_than *upper_limit*

Specifies that only those entries whose attribute value (latency, transition time or skew) is less (more negative) than *upper_limit* are to be shown. You cannot use this option in summary or skew reports.

-slack_lesser_than *slack_upper_limit*

Specifies that only those entries whose slack value is less (more negative) than *slack_upper_limit* are to be shown. For skew report entries slack is the worst slack over all paths launched by the from-pin and captured by the to-pin. For transition time or latency report entries, slack is defined as the worst slack of all paths either launched by a transition at the sink pin (if the *-launch* option is used) or captured by a transition at the sink pin (if the *-capture* option is used). Note that this option is exclusive with -

clock_synthesis_view option.

Note that the use of this filter might greatly increase the runtime of this report. However, when used with the *-capture* option the effect on runtime should be small, since the tool can use cached slack values to avoid expensive recomputations. You cannot use this option in summary reports.

-include_uncertainty_in_skew

Specifies that the user-defined uncertainty between the sequential devices in a launch/capture pair is to be considered a component of skew. You cannot use this option in summary or skew reports.

-verbose

Writes additional detail to the clock timing report. Instead of a single line per pin, verbose reports trace the entire source-to-sink path through a clock network to show how the final reported attribute (skew, latency or transition time) was accumulated over the course of the path. You cannot use this option in summary reports.

-nets

Shows nets in verbose path traces. The default is not to show nets. To show the delay for the nets, use the **-input_pins** option. This option is similar to its counterpart in the **report_timing** command.

-physical

Shows the locations of the pins and the capacitive loads for the pins and nets in verbose path traces. The loads are displayed as a pair of values of which the first is the wire capacitance of the net and the second value is the total capacitance driven by the driver. If the pin location cannot be determined, the cell location is displayed, with the coordinates in microns. This option is similar to its counterpart in the **report_timing** command.

-show_clocks

Shows the launching and capturing clocks for every interclock skew entry in the report. This is useful if *from_clock_list* or *to_clock_list* contain more than one clock each. You cannot use this option in interclock skew reports.

-derate

Specifies that derate factors are to be shown in the report. By default derate factors are not displayed. Derate factors are only shown when the *-verbose* option is specified.

-variation

Reports parametric on-chip variation (POCV) information by expanding column "Incr" to columns "Mean", "Sensit", "Corner" and "Value", and expanding column "Path" to columns "Mean", "Sensit" and "Value". This option works only when POCV analysis is enabled.

-clock_crossing

Specifies that only skews between interacting clock domains (i.e. clocks connected by at least one non-false data path) will be included in the report. This option can only be used in interclock_skew reports.

-clock_synthesis_view

Generates the report based on the way clock synthesis processes the clock networks. By default, this option is false. With this option the generated clocks become part of their master clock's network. Thus, subreports only of the master clocks will be shown, with the clock pins clocked by the respective generated clock becoming part of the pins to be reported in the master clock's report. Clock tree exceptions will be taken into account here. Clock source latency will be excluded here. Note that this option is exclusive with *-slack_lesser_than* option.

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. If this option is not used, most of

the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-significant_digits *digits*

Specifies the number of digits after the decimal point to be displayed for time values in the generated report. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, the tool uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

DESCRIPTION

This command generates a report of clock timing information for the current design.

There are several reporting types provided, which allow you to examine skew, latency and transition time attributes of a given clock network or sub-network at various levels of generality. By default, the report displays the values of these attributes only at sink pins (that is, the clock pins of sequential devices) of the clock network. You use the *-verbose* option to display source-to-sink path traces. If you specify several clock domains, the **report_clock_timing** command generates a separate subreport for each clock domain.

The summary style report is at the highest level of abstraction. This report provides only a list of maxima and minima of the skew, latency, and transition time attributes over the given networks. At a lower level of abstraction are the transition, latency, and skew type reports, called list style reports, in which you can sort, filter and display the worst set of sink pins in the given network with respect to a single attribute of interest. For skew reports, each report entry is a pair of sink pins and their relative skew. For transition time or latency reports, each entry corresponds to a single sink pin. The lowest level of abstraction is provided by verbose mode, which replaces every sink pin in a list style report by a corresponding source-to-sink path trace.

In both summary and list style reports, the rightmost column is an attributed column. Corresponding to each sink pin, the set of character symbols in this column indicates the following:

Symbol	Meaning
r	Rising transition
f	Falling transition
p	Propagated clock to this pin
i	Clock inversion to this pin
-	Launching transition
+	Capturing transition
e	Exception on this pin

In verbose mode, back-annotations on path elements in the timing path are indicated by using a character symbol. For definitions of these character symbols, see the manual page of the **report_timing** command.

Skews reported by **report_clock_timing** are local skews only. Local skew exists from one sink pin to another only as long as their associated sequential devices are connected via a data path in the appropriate from-to sense. Note that even if all data paths between the sequential devices are false because of user-defined exceptions, the skew still qualifies as local skew if the devices are connected topologically.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example shows a typical summary style report.

```
prompt> report_clock_timing -type summary
****************************************
Report : clock timing
        -type summary
        -nworst 1
 ...

****************************************

  Clock: CK1
---------------------------------------------------------------------------
  Maximum setup launch latency:
    flop9/CP                              3.08      rp-+

  Minimum setup capture latency:
    or2_3/B                               1.15      fpi-+

  Minimum hold launch latency:
    or2_3/B                               1.15      fpi-+

  Maximum hold capture latency:
    flop9/CP                              3.08      rp-+

  Maximum active transition:
    or2_3/B                               0.20      fpi-+

  Minimum active transition:
    or2_3/B                               0.09      fpi-+

  Maximum setup skew:
    flop9/CP                                        rp-+
    flop10/CP                              0.87     rp-+

  Maximum hold skew:
    flop9/CP                                        rp-+
    flop10/CP                             -0.21     rp-+
---------------------------------------------------------------------------
```

The following example displays the five worst setup skews in the clock network of CLK1, taking uncertainty into account.

```
prompt> report_clock_timing -clock CLK1 -type skew -setup \
  -nworst 5 -include_uncertainty_in_skew

****************************************
Report : clock timing
        -type skew
        -nworst 5
        -setup
        -include_uncertainty_in_skew
 ...
```

```
****************************************

 Clock: CLK1

 Clock Pin                    Latency   Uncert    Skew
--------------------------------------------------------------------------
 f2_2/CP                       6.11                 rp-+
 f2_1/CP                       2.01     0.11     4.21     fp-+

 l3_2/G                       4.10                  rpi-
 f1_2/CP                       1.00     0.22     3.32     rpi-+

 l2_2/G                       4.11                  rp-
 f1_2/CP                       1.00     0.12     3.23     rpi-+

 f2_2/CP                       6.11                 rp-+
 l3_3/G                       3.01     0.11     3.21     rp-

 l1_3/G                       5.11                  rp-
 f2_1/CP                       2.01     0.11     3.21     fp-+
--------------------------------------------------------------------------
```

The following displays the five worst launching latencies for hold paths in the clock network of CLK2.

```
prompt> report_clock_timing -clock CLK2 -hold -launch -nworst 5 \
  -type latency
****************************************
Report : clock timing
        -type latency
        -launch
        -nworst 5
        -hold
 ...

****************************************

 Clock: CLK2

                              --- Latency ---
 Clock Pin                Trans     Source    Network   Total
--------------------------------------------------------------------------
 f1_2/CP                  0.04      0.00      1.00      1.00      rpi-+
 f1_3/CP                  0.00      0.01      1.00      1.01      fp-+
 f2_1/CP                  0.01      0.01      2.00      2.01      rp-+
 f1_1/CP                  0.00      0.00      3.00      3.00      rpi-+
 f3_2/CP                  0.00      0.00      3.00      3.00      fpi-+
--------------------------------------------------------------------------
```

The following example demonstrates how to request a verbose report showing the worst local skew from f2_2/CP to any other sink pin.

```
prompt> report_clock_timing -type skew -verbose -from f2_2/CP
****************************************
Report : clock timing
        -type skew
```

```
        -verbose
        -nworst 1
        -setup
    ...

    ****************************************

     Clock: CLK1

     Startpoint: f2_2 (rising edge-triggered flip-flop clocked by CLK1)
     Endpoint: f2_1 (rising edge-triggered flip-flop clocked by CLK1)

     Point                    Trans      Incr      Path
     -------------------------------------------------------------------
     clock source latency                        0.11      0.11
     clk2 (in)                0.00       0.00      0.11 r
     az_1/Z (B1I)                0.09       1.00 H    1.11 r
     az_2/Z (B1I)                0.13       1.00 H    2.11 r
     bf2_2_1/Z (B1I)              0.02       1.00 H    3.11 r
     if2_2_1/Z (IVA)              0.44       1.00 H    4.11 f
     bf2_2_2/Z (B1I)              0.01       1.00 H    5.11 f
     if2_2_2/Z (IVA)              0.13       1.00 H    6.11 r
     f2_2/CP (FD1)                0.13       0.00      6.11 r
     startpoint clock latency                          6.11

     clock source latency                        0.01      0.01
     clk2 (in)                0.00       0.00      0.01 r
     az_1/Z (B1I)                0.02       1.00 H    1.01 r
     az_3/Z (B1I)                0.01       1.00 H    2.01 r
     f2_1/CP (FD1)                0.01       0.00      2.01 r
     endpoint clock latency                            2.01
     -------------------------------------------------------------------
     startpoint clock latency                          6.11
     endpoint clock latency                           -2.01
     -------------------------------------------------------------------
     skew                                        4.10
```

The following example traces the two worst launch latencies for hold paths in the clock network of CLK1.

```
prompt> report_clock_timing -type latency -hold -verbose \
                -nworst 2 -clock CLK1
****************************************
Report : clock timing
        -type latency
        -verbose
        -launch
        -nworst 2
        -hold
...

****************************************

 Clock: CLK1

 Endpoint: f1_2 (rising edge-triggered flip-flop clocked by CLK1')

 Point                    Trans      Incr      Path
```

```
  ------------------------------------------------------------------
  clock source latency                    0.00      0.00
  clk1 (in)                    0.00       0.00      0.00 f
  if1_2_1/Z (IVA)                 0.04    1.00 H    1.00 r
  f1_2/CP (FD1)                   0.04    0.00      1.00 r
  total clock latency                             1.00

  Endpoint: f1_3 (rising edge-triggered flip-flop clocked by CLK1)

  Point                       Trans      Incr      Path
  ------------------------------------------------------------------
  clock source latency                    0.01      0.01
  clk1 (in)                    0.00       0.00      0.01 r
  bf1_3_1/Z (B1I)                 0.00    1.00 H    1.01 r
  f1_3/CP (FD1)                   0.00    0.00      1.01 r
  total clock latency                             1.01
```

The following example makes use of a slack filter to display the worst three skews between latches whose latch-to-latch paths are violating.

```
prompt> report_clock_timing -type skew -nworst 3 \
                            -slack_lesser_than 0.0

****************************************
Report : clock timing
        -type skew
        -slack_lesser_than 0.00
        -nworst 3
        -setup
 ...

****************************************

  Clock: CLKA

  Clock Pin                Latency   CRP      Skew      Slack
-------------------------------------------------------------------------------
  f2_2/CP                   6.01                         rp-+
  f2_1/CP                   2.11     -0.03    3.87      -1.49    rp-+

  I2_2/G                   4.01                         rp-
  f1_2/CP                   1.10     -0.21    2.70      -6.64    rpi-+

  I1_3/G                   5.01                         rp-
  f2_1/CP                   2.11     -0.32    2.58      -1.63    rp-+
-------------------------------------------------------------------------------
```

The following requests the two largest setup skews for paths both launched and captured by any clock networks in the list *$my_clocks*.

```
prompt> report_clock_timing -type interclock_skew \
  -from_clock $my_clocks -to_clock $my_clocks \
  -nworst 2 -include_uncertainty_in_skew -show_clocks

****************************************
Report : clock timing
        -type interclock_skew
```

```
        -nworst 2
        -setup
        -include_uncertainty_in_skew
        -show_clocks
    ...

    ****************************************

     Number of startpoint pins   : 907
     Number of endpoint pins     : 907
     Number of startpoint clocks : 5
     Number of endpoint clocks   : 5

     Clock Pin                        Latency   Uncert    Skew
    -------------------------------------------------------------------------------
     orig_clk
     orig_if/ram_pdp1/FFB35/CK          1016.72                 rp-+
     dcd_ram/ram_clk
     dcd_ram/dcd_ram/RAM/CKRB             149.60    195.00    1062.12   rp-+

     comp_clk
     comp_if/c_port_if/edge_trig_reg/CK    1400.42                  rp-+
     comp_clk
     comp_if/c_port_if/posi/iq_reg/CK      1159.09   199.00    440.34    rp-+
    -------------------------------------------------------------------------------
```

SEE ALSO

report_clocks(2)
report_timing(2)
set_clock_uncertainty(2)
shell.common.report_default_significant_digits(3)

report_clock_tree_options

Reports existing target skew/latency constraints for clock trees.

SYNTAX

status **report_clock_tree_options**

DESCRIPTION

This command reports all existing settings like clock target skew/latency constraints, fanout-based ndr, etc, previously defined by **set_clock_tree_options** commands.

EXAMPLES

The following example reports existing clock clock target skew/latency constraints. prompt> **report_clock_tree_options**

SEE ALSO

set_clock_tree_options(2)
remove_clock_tree_options(2)

report_clock_tree_reference_subset

Reports previously defined reference settings for a clock tree.

SYNTAX

status **report_clock_tree_reference_subset**
[-clocks *clock_list*]

Data Types

clock_list collection

ARGUMENTS

-clocks *clock_list*

Reports reference settings for the specified list of clocks.

DESCRIPTION

Use this command to report the specified references settings for clock trees. If the **-clocks** option is used, reference lists of the specified clocks are reported. If the **-clocks** option is not used, all clock tree reference settings will be reported.

EXAMPLES

The following example reports reference cell settings for clock tree clk1.

prompt> **report_clock_tree_references_subset -clocks clk1**

SEE ALSO

remove_clock_tree_reference_subset(2)
set_clock_tree_reference_subset(2)

report_clock_trunk_endpoints

Reports clock trunk endpoints.

SYNTAX

status **report_clock_trunk_endpoints**
[-clock *clock_list*]
[-corners *corner_list*]
[-script]
[-nosplit]
[-significant_digits *digits*]
[*pins_or_ports*]

Data Types

clock_list list or collection
corner_list list or collection
pins_or_ports list or collection

ARGUMENTS

-clock *clock_list*

Report clock trunk endpoints only for the specified clocks. By default, clock trunk endpoints are reported for all clocks.

-corners *corner_list*

Report clock trunk endpoints only for the specified corners. By default, clock trunk endpoints are reported for all corners.

-script

Write out **set_clock_trunk_endpoints** commands that describe the current clock trunk. The commands are written to the console. The **set_clock_trunk_endpoints** commands can by used to re-create the clock trunk endpoints.

-nosplit

Prevents line splitting. This can useful for scripts that extract information from the report. By default, the report generates a new line when the text cannot fit in the alloted space in a column.

-significant_digits *digits*

Specifies the number of digits after the decimal point displayed for time values in the generated report.

pins_or_ports

Report clock trunk endpoints only for the specified pins or ports. By default, clock trunk endpoints are reported for all pins and

ports.

DESCRIPTION

This command reports the phase_delay annotated on the clock trunk endpoints (pins and ports) for specified clock and corners. All endpoints in the current physical hierarchy and child hierarchies below the current physical hierarchy are reported.

For a given clock and corner, the endpoints are sorted in the decreasing order of annotated phase delay. Dominant clock trunk endpoints are marked with a "*" in a separate column as shown in the EXAMPLES section.

By definition, in case a single clock pin is driving more than one trunk endpoint, the endpoints with the worst downstream phase delay is called the "dominant" endpoint.

The term "dominant" is relative to a single clock port of a block. If a block clock port is driving only one trunk endpoint, it will be marked as dominant endpoint. A block may have more than one "dominant" trunk endpoints.

The worst downstream phase_delay will be different for different corners. So an endpoint that is "dominant" in one corner may not be "dominant" in other corner.

Multicorner-Multimode Support

EXAMPLES

The following example reports all clock trunk endpoints.

```
prompt> report_clock_trunk_endpoints
****************************************
Report : report_clock_trunk_endpoints
Design : TOP_DG
Version:
Date   :
****************************************

Endpoint             Phase Delay  Dominant Pt  Clock   Corner   Block
--------------------------------------------------------------------------------
CPU_011/TOP_ff_U308/CK    0.40        *        clock1  corner0  CPU_011
CPU_011/TOP_1_0_ff/CK     0.30                 clock1  corner0  CPU_011
```

SEE ALSO

gui_load_clock_trunk_planning(2)
remove_clock_trunk_endpoints(2)
set_clock_trunk_endpoints(2)
synthesize_clock_trunk_endpoints(2)
synthesize_clock_trunks(2)

report_clock_trunk_qor

Generates summary of timing information of the design for QoR evaluation of clock trunk planning (CTP).

SYNTAX

status **report_clock_trunk_qor**
[-clock *clocks*]
[-endpoint_limit *endpoint_limit*]
[-from_block *block_instances*]
[-to_block *block_instances*]
[-sort_by wns | wns_ocv]
[-significant_digits *digits*]
[-nosplit]
[-violating_only]

Data Types

clocks list
endpoint_limit integer
block_instances list
digits integer

ARGUMENTS

-clock *clocks*

Specifies the clocks for which the report is to be generated. If you do not specify this option, the command generates report for all clocks of currently active modes.

-endpoint_limit *endpoint_limit*

Specifies the limit for the total number of end-points for specified fanout limit and block fanout limit values. The command exits with an error message if this limit is exceeded; this might be an indication of an issue with driving cell specification for blocks. The default value of this option is 100.

-from_block *block_instances*

Reports only the paths that originate from the specified block instances. By default, the command considers paths originating from all blocks.

-to_block *block_instances*

Reports only the paths that terminate at the specified block instances. By default, the command considers paths terminating at all blocks.

-sort_by wns | wns_ocv

Specifies the criteria for sorting the output report. By default, the output is sorted based on worst negative slack (WNS) without taking into account the degradation due to OCV.

-significant_digits *digits*

Specifies the number of significant digits (digits to the right of the decimal point) to report. Allowed values are from 0 through 13. Using this option overrides the value set by the **shell.common.report_default_significant_digits** application option. By default, the number of significant digits is 2.

-nosplit

Prevents wide report lines from being split. Most of the information is listed in fixed-width columns. The **-nosplit** option prevents line-splitting and facilitates writing scripts to extract information from the report output. By default, if the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-violating_only

Indicates that only violating paths are to be reported in the output.

DESCRIPTION

This command generates a summary report of timing information for clock trunk planning (CTP). Use the report to evaluate the QoR for specific clocks or all clocks.

The first time you run **report_clock_trunk_qor**, the command might trigger a full or an incremental timing update if required. Subsequent runs of this command will not trigger a timing update if there are no netlist changes or a changes in timing constraints.

EXAMPLES

The following example reports endpoint, WNS, TNS, and other information for all clocks that belong to the active modes of the design.

```
prompt> report_clock_trunk_qor
****************************************
Report : report_clock_trunk_qor
Design :
Version:
Date   :
****************************************

Clock    Trunk End Point (Launch)  Trunk End Point (Capture)  Divergence Point
  WNS (Ideal Clock)  OCV       CRP       WNS       TNS
    Number of Paths   Violating Paths  Timing Start Point  Timing End Point
-------------------------------------------------------------------------------
CLK      instD/clk1              instE/clk              b11/Y
  1.22          0.06      0.02      1.18      0.00
                2             0  instD/f1/D [D]     instE/f2/D [E]
CLK      instC/clk              instD/b1/A             b12/Y
  1.41          0.07      0.02      1.36      0.00
                1             0  instC/f1/D [C]     instD/f1/D [D]
```

The following example reports paths - launch and captured by clock CLK and originating from block B1

```
prompt> report_clock_trunk_qor -clock [get_clocks CLK] \
  -from_block [get_cells B1]
```

SEE ALSO

synthesize_clock_trunks(2)
shell.common.report_default_significant_digits(3)

report_clocks

Reports clock-related information.

SYNTAX

string **report_clocks**
[-attributes]
[-skew]
[-groups]
[-nosplit]
[-significant_digits *digits*]
[-modes *mode_list*]
[*clock_list*]

Data Types

digits int
mode_list list
clock_list list

ARGUMENTS

-attributes

Shows clock attributes and provides a list of all the clocks in the current design. The information for each clock includes source type, signal rise and fall times, and attributes. This report is shown by default.

-skew

Reports clock latency (source and network latency) and uncertainty information set by the **set_clock_latency** and **set_clock_uncertainty** commands, respectively.

Clock network latency information includes rise latency and fall latency. Clock source latency information includes rise latency and fall latency for early and late arrivals. Clock uncertainty information includes intraclock setup or hold uncertainty and interclock setup or hold uncertainty. This option also reports any fixed clock transition set by using the **set_clock_transition** command.

-groups

Shows the current setting for clock groups, including grouping of exclusive clocks and asynchronous clocks set by using the **set_clock_groups** command.

-nosplit

Retains long lines in the output, even if a column overflows. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This

option prevents line-splitting and facilitates writing scripts to extract information from the report output.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option to override the default.

This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, the tool uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

-modes *mode_list*

Specifies the list of modes to be used in the report. A subreport is produced for each clock in the mode when the *mode_list* is specified.

clock_list

Lists the clocks that must be reported. Substitute the list you want for *clock_names*.

DESCRIPTION

Reports clock-related information. Displays all clock-related information on a design. The report contents are controlled by the options used. If you do not specify any options, the command displays the attributes report.

When a clock is created, it is added to the other clocks list if some clocks are defined to be exclusive or asynchronous with any other clocks in the design. When a clock is removed, it is removed from its related lists. To see if a clock is active or not, use the **report_clocks** command with the **-attributes** option.

Some information, such as the waveform of a generated clock or propagated skew, is displayed after a timing update. This can result from manually running **update_timing** or an implicit timing update as a result of executing other commands. This behavior is intended to help you to define all clocks without incurring the cost of a complete timing update.

To obtain a list of all clocks in the design, use the **all_clocks** command.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-modes** option.

EXAMPLES

```
prompt> create_clock -period 20.000000 \
[get_ports {CLK}]

prompt> report_clocks

****************************************
Report : clock
Design : counter
Version:
Date   :
****************************************
```

```
Attributes:
   p - propagated_clock
   G - Generated clock

Clock        Period   Waveform          Attrs    Sources
-------------------------------------------------------------------------
CLK          20.00   {0 10}                  {CLK}

prompt> set_clock_latency 2.4 [get_clocks CLK]
prompt> set_clock_latency 0.17 -source -early [get_clocks CLK]
prompt> set_clock_latency 0.19 -source -late  [get_clocks CLK]
prompt> set_clock_uncertainty 1.3 [get_clocks CLK]
prompt> set_clock_transition  1.7 [get_clocks CLK]
prompt> report_clocks -skew

****************************************
Report : clock_skew
Design : counter
Version:
Date   :
****************************************

            Rise         Fall         Hold         Setup
Object       Delay        Delay       Uncertainty    Uncertainty
-------------------------------------------------------------------------
CLK          2.40         2.40         1.3          1.3

-------------------------------------------------------------------------
                        Source Latency
-------------------------------------------------------------------------
Object      Min Rise      Min Fall      Max Rise      Max Fall
-------------------------------------------------------------------------
CLK          0.17         0.17         0.19          0.19

            Rise         Fall
Object       Transition    Transition
-------------------------------------------------------------------------
CLK          1.7          1.7

prompt> set_clock_latency -late -source -dynamic 0.5 4.5 [get_clocks clk]
prompt> set_clock_latency -early -source 2.5 -dynamic 0.5 [get_clocks clk]
prompt> report_clocks -skew

****************************************
Report : clock_skew
Design : latency
Version:
Date   :
****************************************

            Min Condition Source Latency     Max Condition Source Latency
-------------------------------------------------------------------------------------
Object      Early_r Early_f  Late_r  Late_f   Early_r Early_f  Late_r  Late_f  Rel_clk
-------------------------------------------------------------------------------------
clk (static)    2.00    2.00    4.00    4.00     2.00    2.00    4.00    4.00  --
clk (dynamic)   0.50    0.50    0.50    0.50     0.50    0.50    0.50    0.50  --
          -------------------------------------------------------------------------
```

```
clk (total)     2.50   2.50   4.50   4.50     2.50   2.50   4.50   4.50  --

prompt> set_clock_groups -ex -group {clk1 clk3}
prompt> set_clock_groups -asyn -name a1 -group clk2 -group clk2
prompt> set_active_clocks {clk1 clk2}
prompt> report_clocks -groups

****************************************
Report : clock_groups
Design : test
Version:
Date   : Thu May  9 13:13:51 2002
****************************************

Active clocks:
  clk1  clk2

Total exclusive groups: 1.
NAME : clk1_others
        -group {clk1 clk3 }
        -group {clk2 clk4 }

Total asynchronous groups: 1.
NAME : a1
        -group {clk2 }
        -group {clk4 }
```

SEE ALSO

all_clocks(2)
create_clock(2)
remove_clock_groups(2)
set_clock_groups(2)
set_clock_latency(2)
set_clock_transition(2)
set_clock_uncertainty(2)
shell.common.report_default_significant_digits(3)

report_collection

Output a tabular report of attribute values for elements in a collection.

SYNTAX

report_collection
collection
[-type *report_type*]
[-header *header_type*]
[-max_rows *value_count*]
[-nopslit]
[-columns *column_specification*]

string *collection*
list *report_type (*
string *header_type*
int *value_count*
list *column_specificaton*

ARGUMENTS

collection

Specifies the collection on which to report. The collection may be homogeneous or heterogeneous. The report comprises a header with information about the report such as the application name and version and the date and a tabular report on attribute values for elements of the collection. The tabular report content is determined by the of report requested with the *-type* option. The attributes on which the report is generated is determined by the argument to the *-columns* option.

-type *report_type*

This option accepts an argument of a list of valid report type names. A report type is one of the following values: "values" | "statistics" | "distribution". If the option is omitted, the default report type is "values". The report types may be combined in a list to output multiple reports in one run. Each invocation of the command produces a single report header output unless it is suppressed with the *-header* option.

The "values" report is a tabular output of attribute values for the collection elements, with one row of values per collection element. The report columns comprises formatted attribute values for the objects.

The "statistics" report type outputs the following statistical data about the attribute values

* Minimum, lower quartile, median upper quartile, and maximum values for numeric attributes. * The mean and standard deviation for numeric attributes. * Number of null attribute values for each attribute. * Number of valid attribute values for each attribute. * Number of unique attribute values for each attribute.

The "distribution" report type outputs a header section with information about the report, followed by the following statistical values about the data.

* Number of null attribute values for each attribute. * Number of valid attribute values for each attribute. * Number of unique attribute values for each attribute. * Distribution of values with number of occurrences of each attribute value..

If both "statistics" and "distribution" reports are requested, the overlapping portion of the two reports is output only once.

All report types begin with a line displaying the collection composition including the size of the input collection followed by object count by object class type. When "statistcs" or "distribution" type report is included, the object count breakdown by object class type is always provided regardless of the size of the input collection. In a "values" only report, however, breakdown of object count by object class type is provided for a heterogeneous collection only when the collection contains one hundred or fewer items.

-header *header_type*

This option accepts one of the following valid argument values: "standard" | "none". If the option is omitted, the default header type is "standard". The "standard" header consists of the report name, command arguments, product name, product version, and the report date.

-max_rows *value_count*

This option indicates how many rows of data to include in the values and the value distribution tables. If the **collection** is very large, you may want to use the **-max_rows** option to limit the size of your report output. Number of text lines in the report may be greater than the *-max_rows* argument value if some data rows require multiple lines to output. This option value has no impact on the distribution and statistics calculations, for which the entire collection is processed. However, the **-max_rows** value, if given will limit the number of rows output for the value distribution table. If **-max_rows** value is given and truncates either the values or the distribution table output, a line is added to show the number of remaining items that were not shown.

-nosplit

This option indicates that the default column formatting used for a value is that is too long for the column width is "none", rather than the default "split". See the -columns option description below for more details.

-columns *column_specification*

Indicates the set of attributes on which to report and their order in the report. At a minimum, *column_specification* is a list of attribute names of elements of the collection.

The special attribute name "object_class" may be used to include the element object class names, such as "cell" or "net", in the report output.

If the option is omitted, then the object full names and object class (or type) names are output by default.

Each attribute name may be followed by a list of token-value pairs to indicate report column formatting directives. The value tokens for formatting are *width*, *precision*, *justify*, *label*, and *fit*. All column formatting specifications are optional.

width token is followed by a positive integer to value indicate the desired width of the column in number of characters. If not given, then the attribute value is queried for the first 100 elements of the collection and the longest value string determines the width of the column.

precision token is followed by a positive integer to value indicate the desired precision used to display floating point values (digits after the decimal point). If not given then the default precision for the attribute is used. This is ignored for non floating point attributes.

justify is followed by one of "left" or "right" to indicate the value's alignment in the column. If not given, then numeric attributes are justified right and other attributes are justified left.

By default, the column headers show the name of the attribute. *label* is followed a string to display in the report column header in place of the attribute name.

fit is followed by one of the following valid argument values: "split" | "none" | "truncate" | "wrap" | "elide_left" | "elide_center" | "elide_right". This option indicates how to format an attribute value that is too long to fit in the column's width. If option is omitted, the default formatting is "split". However, if the option *-nosplit* is given, then the default formatting becomes "none". Explicit *fit* values given in a column specification overrides the default set by *-noSplit*.

The "split" formatting inserts a newline after the value which is too long and continues the tabular row on the next line, justifying the next value to the correct column.

The "none" formatting does nothing to reformat a value that is too long, continuing with the next column value. Therefore, the remainder of the tabular row will not be aligned with column headers.

The "truncate" formatting truncates the value to the width of the column, discarding the remainder of the value.

The "wrap" formatting truncates the value to the column and continues the rest of the tabular row on the same row. The remainder of the value is output on the following line(s), taking as many lines as is needed to finish outputting the rest of the value.

The "elide_left" formatting truncates the beginning of the value and adds the elision string "..." at the beginning of the value.

The "elide_center" formatting omits the middle of the value and inserts the elision string "..." at the center of the value.

The "elide_right" formatting truncates the end of the value and adds the elision string "..." at the end of the value.

DESCRIPTION

This command outputs a tabular report of attribute values for elements of the given collection. The attribute values in the report are determined by the list of attributes given as the **-columns** option argument. The columns in the tabular report will be ordered by order of attributes in the list. The command **list_attributes** may be called to see a list of attributes defined for an object class.

The **-column** option also accepts additional optional column formatting specifications in the attribute list. You may specify the header label and the width for each column, how to justify the value, and how to fit a value in a column when the value is too long to fit within the column width.

If the collection is large, the report size may be limited by indicating a **-max_rows** value. The commands **filter_collection** and **sort_collection** may be called to filter and sort a collection based on some criteria prior to creating a report for the collection elements.

Statistical and value distribution reports on the collection elements may be produced using the *-type* option arguments "statistics" and "distribution", respectively. Report types may be combined in a list.

The report header is followed by a line of report with collection size and object count by object data types. Precise object counts by types are always shown for "statistics" and "distribution" type reports. For "values" report, it is shown if the collection is homogenous or small. If the collection is heterogeneous and large, the object type shown for "values" report is "heterogeneous".

EXAMPLES

The following example from IC Compiler II creates a statistical report on a collection of cells along with a value distribution output.

```
icc2_shell> set cells [get_cells -hierarchical *]
icc2_shell> report_collection $cells -type {statistics distribution}
```

The following example, also from IC Compiler II, creates a report on a collection of cells with the full_name, area, and number_of_pins attribute values.

```
icc2_shell> set cells [get_cells U*]
icc2_shell> report_collection $cells -columns {full_name area number_of_pins}
```

The next example creates the same report but limits the width of the full_name column to 26 characters, designating "elide_center" formatting for full_name values that are longer than 26 characters.

```
icc2_shell> report_collection $cells -columns \
                    {{full_name {width 26} {fit elide_center}} \
                     area number_of_pins}
```

The next examples limits the output to the first 10 objects in the collection.

```
icc2_shell> report_collection $cells -columns \
                    {{full_name {width 26} {fit elide_center}} \
                     area number_of_pins} -max_rows 10
```

The next example first sorts the original collection of cells by the area attribute value in descending order, limiting the resulting collection to the top 10 area values. The example then careates a report for the resulting collection.

```
icc2_shell> set sorted_cells [sort_collection -dictionary \
                   $cells {area- full_name} -limit 10]
icc2_shell> report_collection $sorted_cells -columns \
                    {{full_name {width 26} {fit elide_center}} \
                     area number_of_pins}
```

The next example reports on a sorted collection as in the above example but further limits the report to the first 10 objects in the collection. Note that this command may produce a report that is different from the above example. Limiting the sort to the first 10 values may have produced a collection with more than 10 objects since multiple objects may share the same area value.

```
icc2_shell> set sorted_cells [sort_collection -dictionary \
                   $cells {area- full_name} -limit 10]
icc2_shell> report_collection $sorted_cells -columns \
                    {{full_name {width 26} {fit elide_center}} \
                     area number_of_pins} -max_rows 10
```

SEE ALSO

collections(2)
filter_collection(2)
sort_collection(2)
write_collection(2)
list_attributes(2)

report_congestion

Reports the congestion statistics.

SYNTAX

status **report_congestion**
[-mode summary | global_route_cell_edge_based]
[-boundary *bbox* | *bbox_list*]
[-layers *collection_of_layers*]
[-rerun_global_router]
[-overflow_threshold *threshold*]
[-include_soft_congestion_maps]
[-snap_cells]
[-nosplit]
[-significant_digits *digits*]

Data Types

bbox rectangle
bbox_list list of rectangles
collection_of_layers collection of layers
threshold int
digits int

ARGUMENTS

-mode summary | global_route_cell_edge_based

Specifies the mode for reporting the congestion map.

The following values are valid:

- **summary**
  Generates a global route congestion summary report that provides information about total overflow, max overflow, number of global route cells with overflow, and number of global route cells with max overflow.
- **global_route_cell_edge_based**
  Generates a global route cell edge based congestion report that provides information about all queried global route cells in terms of layer name, bounding box, demand, and capacity.

-boundary *bbox* | *bbox_list*

Generates a congestion report for only the specified region.

If you do not specify this option, the tool generates a congestion report for the entire block.

-layers *collection_of_layers*

Generates a congestion report for only the specified layers.

If you do not specify this option, the tool generates a congestion report for all routing layers.

-rerun_global_router

Rerun congestion map only mode global router to generate latest congestion information.

If you do not specify this option, the tool reports the congestion information got from the database.

-overflow_threshold *threshold*

Reports global route cells with overflow larger than or equal to this threshold. The threshold is checked against each global route cells per layer. This option must be used with **-mode global_route_cell_edge_based**.

If you do not specify this option, the tool reports global route cells regardless of the overflow value.

-include_soft_congestion_maps

Includes existed soft congestion maps in the report. To specify if the tool exports the soft congestion maps into the database, refer to application option **route.global.export_soft_congestion_maps**.

If you do not specify this option, the tool skips the soft congestion map reporting.

-snap_cells

Snaps cells to closest cell row before running global router and restores the original placement afterward.

-nosplit

Does not split lines if column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13; the default is determined by the **shell.report_default_significant_digits** application option, whose default is 2. Use this option to override the default.

DESCRIPTION

This command reports the congestion statistics for the current block.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **report_congestion** command:

```
prompt> report_congestion
****************************************
Report : congestion
```

```
Design : r4000
Version:
Date   : Tue Jul  7 01:47:18 2015
****************************************

Layer    |   overflow    |           # GRCs has
Name     | total  |  max  | overflow (%)     | max overflow
-----------------------------------------------------------------
Both Dirs |      1 |    1 |      1  ( 0.03%) |      1
H routing |      1 |    1 |      1  ( 0.03%) |      1
V routing |      0 |    0 |      0  ( 0.00%) |      0
```

The following example runs the **report_congestion** command in global route cell edge based mode:

```
prompt> report_congestion -mode global_route_cell_edge_based \
      -boundary {{$llx1 $lly1} {$urx1 $ury1}}
****************************************
Report : congestion
Design : r4000
Version:
Date   : Mon Jul  6 20:18:14 2015
****************************************
Layer   Boundary                    Demand/Capacity
M1      {{0 0} {2.52 2.52}}          0/8
M2      {{0 0} {2.52 2.52}}          0/7
M3      {{0 0} {2.52 2.52}}          0/8
M4      {{0 0} {2.52 2.52}}          0/8
M5      {{0 0} {2.52 2.52}}          0/8
M6      {{0 0} {2.52 2.52}}          0/8
M7      {{0 0} {2.52 2.52}}          0/2
```

The following example runs the **report_congestion** command with overflow threshold set to 1.

```
prompt> report_congestion -mode global_route_cell_edge_based -overflow_threshold 1
****************************************
Report : congestion
Design : r4000
Version:
Date   : Wed Oct 26 01:46:13 2016
****************************************
Layer    Boundary                        Demand/Capacity
M1       {{12.6 105.84} {15.12 108.36}}       1/0
M1       {{15.12 88.2} {17.64 90.72}}        1/0
M1       {{65.52 120.96} {68.04 123.48}}      2/1
M1       {{70.56 113.4} {73.08 115.92}}       1/0
M1       {{75.6 105.84} {78.12 108.36}}       1/0
M1       {{136.08 5.04} {138.6 7.56}}        1/0
```

The following example runs the **report_congestion** command including reporting soft congestion maps.

```
prompt> report_congestion -include_soft_congestion_maps
****************************************
Report : congestion
Design : r4000
Version:
```

```
Date   : Thu Apr 13 00:03:26 2017
****************************************

Hard Congestion Map

Layer     |    overflow     |            # GRCs has
Name      | total  |  max  | overflow (%)      | max overflow
-------------------------------------------------------------------
Both Dirs |     541 |     9 |    211  ( 3.02%) |       2
H routing |      55 |     2 |     33  ( 0.94%) |      22
V routing |     486 |     9 |    178  ( 5.09%) |       2


Accumulated to Soft Level High Congestion Map

Layer     |    overflow     |            # GRCs has
Name      | total  |  max  | overflow (%)      | max overflow
-------------------------------------------------------------------
Both Dirs |    3514 |    18 |    1071  (15.31%) |       1
H routing |     927 |     8 |     307  ( 8.78%) |      14
V routing |    2587 |    18 |     764  (21.84%) |       1


Accumulated to Soft Level Low Congestion Map

Layer     |    overflow     |            # GRCs has
Name      | total  |  max  | overflow (%)      | max overflow
-------------------------------------------------------------------
Both Dirs |    6919 |    21 |    3183  (45.50%) |       1
H routing |    2331 |    10 |    1256  (35.91%) |      14
V routing |    4588 |    21 |    1927  (55.09%) |       1
```

SEE ALSO

route_global(2)
route.global.export_soft_congestion_maps(3)

report_constant_registers

Reports registers that were removed because their values were not used.

SYNTAX

string **report_constant_registers**

DESCRIPTION

Displays a list of registers that were removed from the design by the **compile** command because they could be implemented as a constant. Constant register removal is controlled by the **compile.seqmap.remove_constant_registers** application variable. A register may be reported as implemented using constant 0, constant 1, or X.

EXAMPLES

The following example shows a sample report:

```
****************************************
Report : constant_registers
 ...

****************************************

Register                                  Value
------------------------------------------------------------------------ ------
z0_reg                                    0
z1_reg                                    1
z2_reg                                    X
1
```

SEE ALSO

compile(2)
compile.seqmap.remove_constant_registers(3)

report_constraint

Displays constraint-related information about a design.

SYNTAX

status **report_constraints**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-all_violators]
[-verbose]
[-max_capacitance]
[-min_capacitance]
[-max_transition]
[-max_delay]
[-min_delay]
[-min_pulse_width]
[-min_period]
[-nosplit]
[-significant_digits *digits*]

Data Types

mode_list collection
corner_list collection
scenario_list collection
digits integer

ARGUMENTS

-modes *mode_list*

Specifies the modes to be reported. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is reported separately. If this option is not given, the command reports scenarios of the current mode. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners to be reported. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is reported separately. If this option is not given, the command reports scenarios of the current corner. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios to be reported. Each of the given active scenarios is reported. It is an error to give this option with the **-modes** or **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any).

-all_violators

Displays a summary of all of the optimization and design rule constraints with violations in the current design. The **-verbose** option provides detailed information about constraint violations. Multiple violations for a given constraint are listed from the most violations to the least violations.

-verbose

Displays more detail about constraint calculations.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. The *digits* value must be between 0 and 13. The default value is 2. This option overrides the value set by the **shell.common.report_default_significant_digits** application option.

-max_capacitance

Displays only the max_capacitance constraint information. The max_capacitance constraint is a design rule used to limit total capacitance on a net. The default is to display all optimization and design rule constraints.

-min_capacitance

Displays only the min_capacitance constraint information. The min_capacitance constraint is a design rule used to limit total capacitance on a net. The default is to display all optimization and design rule constraints.

-max_transition

Displays only the max_transition constraint information. The max_transition constraint is a design rule used to limit transition time on a net. The default is to display all optimization and design rule constraints. If the library uses the cmos2 delay model, this option shows the max_edge_rate information.

-max_delay

Indicates that only maximum delay and setup information is to be displayed.

-min_delay

Indicates that only minimum delay and hold information is to be displayed.

-min_pulse_width

Indicates that only minimum pulse width constraint information is to be displayed. similar to the **report_min_pulse_width** command. Using this option will trigger additional timing update and may have some runtime penalty.

-min_period

Indicates that only minimum period constraint information is to be displayed. similar to the **report_min_period** command. Using this option will trigger additional timing update and may have some runtime penalty.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_constraints** command displays the following information for the constraints on the current design:

- Whether the constraint was violated or met
- By how much the constraint value was violated or met
- The design object that was the worst violator.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example shows brief constraint information for the current design:

```
prompt> report_constraints

****************************************
Report : constraint
Design : counter
****************************************

                                  Weighted
Group (min_delay/hold)     Cost    Weight    Cost    Scenario
-------------------------------------------------------------------------
**default**              0.00     1.00     0.00     m1::c1
**async_default**        0.00     1.00     0.00     m1::c1
**clock_gating_default**  0.00     1.00     0.00     m1::c1
clk                    0.00     1.00     0.00     m1::c1
-------------------------------------------------------------------------
min_delay/hold                          0.00

                                  Weighted
Group (max_delay/setup)     Cost    Weight    Cost    Scenario
-------------------------------------------------------------------------
**default**              0.00     1.00     0.00     m1::c1
**async_default**        0.00     1.00     0.00     m1::c1
**clock_gating_default**  0.00     1.00     0.00     m1::c1
clk                    0.35     1.00     0.35     m1::c1
-------------------------------------------------------------------------
max_delay/setup                         0.35

Constraint                            Cost
---------------------------------------------------
min_delay/hold                         0.00  (MET)
max_delay/setup                        0.35  (VIOLATED)
max_transition                         0.00  (MET)
max_capacitance                        0.00  (MET)
```

```
    min_capacitance                 0.00  (MET)
1
```

SEE ALSO

report_timing(2)
report_min_pulse_width(2)
report_min_period(2)

report_constraint_groups

Reports the constraint groups in the design.

SYNTAX

status **report_constraint_groups**
[-type *constraint_type*]
[-count]
[-nosplit]
[*constraint_group_list*]

Data Types

constraint_type string
constraint_group_list string or collection

ARGUMENTS

-type *constraint_type*

Specifies the type of constraint groups to be reported. The *constraint_type* argument is one of the following keywords:

- bus_style
- differential_group
- differential_pair
- matched_wire
- net_priority
- reliability_verification
- resistance_limit
- river_style
- shielding
- wire_length_limit

By default, all constraint types are reported.

-count

Displays only the count of constituent objects for a constraint group. If not specified, then all constituent objects are reported.

-nosplit

Does not split lines if column overflows.

constraint_group_list

Specifies a list of constraint groups to report. The list can contain constraint group names, patterns, or collections. A collection can be specified by using the **get_constraint_groups** command.

DESCRIPTION

This command reports the user-specified constraint groups in the design, as specified by the *create_wire_matching*, *create_length_limit*, *create_differential_group*, *create_net_shielding*, *create_net_priority* and *create_bus_routing_style* commands. By default this command reports all constraint groups but it can be controlled with *-type* or passing appropriate constraint groups collection.

The report includes constraint group name, its type and constituent objects. If *-count* is used then only the count of constituent objects is reported.

EXAMPLES

The following example reports the constraint group *bus_style_7*:

```
prompt> report_constraint_groups bus_style_7

****************************************
Report : report_constraint_groups
Design : ORCA
Version:
Date   :
****************************************

------------------------------------------------------------
Name               Type            Objects
------------------------------------------------------------
bus_style_7          bus_style         prst_n preq_n
```

The following example reports the bus_style type of constraint groups:

```
prompt> report_constraint_groups -type bus_style

****************************************
Report : report_constraint_groups
Design : ORCA
Version:
Date   :
****************************************

------------------------------------------------------------
Name               Type            Objects
------------------------------------------------------------
```

```
bus_style_7          bus_style        prst_n preq_n
bus_style_8          bus_style        prst_n preq_n
```

The following example reports all constraint groups:

```
prompt> report_constraint_groups

****************************************
Report : report_constraint_groups
Design : ORCA
Version:
Date   :
****************************************


------------------------------------------------------------
Name                 Type             Objects
------------------------------------------------------------
matched_wire_0         matched_wire       prst_n preq_n
matched_wire_1         matched_wire       prst_n preq_n
matched_wire_2         matched_wire       prst_n preq_n
matched_wire_3         matched_wire       prst_n preq_n
differential_pair_4     differential_pair   prst_n preq_n
shielding_5           shielding          prst_n preq_n
net_priority_6         net_priority       prst_n preq_n
bus_style_7           bus_style          prst_n preq_n
bus_style_8           bus_style          prst_n preq_n
```

The following example reports all constraint groups but only the count of constituent objects:

```
prompt> report_constraint_groups -count

****************************************
Report : report_constraint_groups
Design : ORCA
Version:
Date   :
****************************************


-----------------------------------------------------
Name                 Type             Count
-----------------------------------------------------
matched_wire_0         matched_wire       2
matched_wire_1         matched_wire       2
matched_wire_2         matched_wire       2
matched_wire_3         matched_wire       2
differential_pair_4     differential_pair   2
shielding_5           shielding          2
net_priority_6         net_priority       2
bus_style_7           bus_style          2
bus_style_8           bus_style          2
```

SEE ALSO

create_bus_routing_style(2)

create_differential_group(2)
create_length_limit(2)
create_net_priority(2)
create_net_shielding(2)
create_wire_matching(2)
get_constraint_groups(2)

report_constraint_mapping_file

Reports the constraint file map for blocks in the design.

SYNTAX

status **report_constraint_mapping_file**
[-design *design*]
[-constraint_type *type*]

Data Types

design design object
type string

ARGUMENTS

-design *design*

Specifies the design which to report. When this option is used together with **-constraint_type**, the command returns the full path to the constraint file of the specified constraint type for the specified design. If **-constraint_type** is not specified, the tool reports all constraint types for the specified design.

-constraint_type *type*

Specifies the type of constraint to report. When this option is used together with **-design**, the command returns the full path to the constraint file of specified type for the specified design. If **-design** is not specified, the tool reports the specified constraint types for all designs.

DESCRIPTION

This command reports the constraint mapping that was read by the *set_constraint_mapping_file* command. Each line of the report shows a specific type of constraint file for a particular block as follows:

block_design_name constraint_type file_name

block_design_name is the reference block name for either the top-level design or a block-level module.

constraint_type is one of the following keywords: **DEF**, **UPF**, **ETM_UPF**, **SDC**, **BUDGET**, **CLKNET**, **PG_CONSTRAINT**, **COMPILE_PG**, **CTS_CONSTRAINT**, **BTM**, or **FLOORPLAN**, **SCANDEF**.

file_name is the constraint file name with its UNIX path.

When no arguments are specified, the tool reports the entire constraint mapping information in the design.

When only the **-design** argument is specified, the tool reports all the constraint files for this design.

When only the **-constraint_type** argument is specified, the tool reports the constraints files for all the designs for the specified type of constraint.

When both the **-design** and **-constraint_type** arguments are specified, the tool reports and returns the full path name to the constraint file that corresponds to the specified design and constraint type.

EXAMPLES

The following example specifies the DEF and UPF constraint files. The files are used in the hierarchical flow from Verilog input to block shaping. The design has a top-level and two child blocks: BLK_A and BLK_B. The content of the constraint mapping file *./map_file.1* is:

```
BLK_A  DEF ./constraints/BLK_A.def
BLK_A  UPF ./constraints/BLK_A.upf
BLK_B  DEF ./constraints/BLK_B.def
BLK_B  UPF ./constraints/BLK_B.upf
TOP    DEF ./constraints/TOP.def
TOP    UPF ./constraints/TOP.upf
```

The following example specifies a constraint mapping file and reports the settings with *report_constraint_mapping_file*.

```
prompt> set_constraint_mapping_file ./map_file.1
prompt> report_constraint_mapping_file
BLK_A DEF /disk1/constraints/BLK_A.def
BLK_A UPF /disk1/constraints/BLK_A.upf
BLK_B DEF /disk1/constraints/BLK_B.def
BLK_B UPF /disk1/constraints/BLK_B.upf
TOP DEF /disk1/constraints/TOP.def
TOP UPF /disk1/constraints/TOP.upf
```

The following example returns the UPF file name for BLK_A.

```
prompt> set UPF_for_A [report_constraint_mapping_file -design BLK_A -constraint_type UPF]
prompt> echo $UPF_for_A
/disk1/constraints/BLK_A.upf
```

SEE ALSO

set_constraint_mapping_file(2)
update_constraint_mapping_file(2)

report_constraints

Displays constraint-related information about a design.

SYNTAX

status **report_constraints**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-all_violators]
[-verbose]
[-max_capacitance]
[-min_capacitance]
[-max_transition]
[-max_delay]
[-min_delay]
[-min_pulse_width]
[-min_period]
[-nosplit]
[-significant_digits *digits*]

Data Types

mode_list collection
corner_list collection
scenario_list collection
digits integer

ARGUMENTS

-modes *mode_list*

Specifies the modes to be reported. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is reported separately. If this option is not given, the command reports scenarios of the current mode. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners to be reported. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is reported separately. If this option is not given, the command reports scenarios of the current corner. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios to be reported. Each of the given active scenarios is reported. It is an error to give this option with the **-modes** or **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any).

-all_violators

Displays a summary of all of the optimization and design rule constraints with violations in the current design. The **-verbose** option provides detailed information about constraint violations. Multiple violations for a given constraint are listed from the most violations to the least violations.

-verbose

Displays more detail about constraint calculations.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. The *digits* value must be between 0 and 13. The default value is 2. This option overrides the value set by the **shell.common.report_default_significant_digits** application option.

-max_capacitance

Displays only the max_capacitance constraint information. The max_capacitance constraint is a design rule used to limit total capacitance on a net. The default is to display all optimization and design rule constraints.

-min_capacitance

Displays only the min_capacitance constraint information. The min_capacitance constraint is a design rule used to limit total capacitance on a net. The default is to display all optimization and design rule constraints.

-max_transition

Displays only the max_transition constraint information. The max_transition constraint is a design rule used to limit transition time on a net. The default is to display all optimization and design rule constraints. If the library uses the cmos2 delay model, this option shows the max_edge_rate information.

-max_delay

Indicates that only maximum delay and setup information is to be displayed.

-min_delay

Indicates that only minimum delay and hold information is to be displayed.

-min_pulse_width

Indicates that only minimum pulse width constraint information is to be displayed. similar to the **report_min_pulse_width** command. Using this option will trigger additional timing update and may have some runtime penalty.

-min_period

Indicates that only minimum period constraint information is to be displayed. similar to the **report_min_period** command. Using this option will trigger additional timing update and may have some runtime penalty.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_constraints** command displays the following information for the constraints on the current design:

- Whether the constraint was violated or met
- By how much the constraint value was violated or met
- The design object that was the worst violator.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example shows brief constraint information for the current design:

```
prompt> report_constraints

****************************************
Report : constraint
Design : counter
****************************************

                                  Weighted
Group (min_delay/hold)     Cost    Weight    Cost    Scenario
-------------------------------------------------------------------------
**default**              0.00     1.00     0.00     m1::c1
**async_default**        0.00     1.00     0.00     m1::c1
**clock_gating_default**  0.00     1.00     0.00     m1::c1
clk                    0.00     1.00     0.00     m1::c1
-------------------------------------------------------------------------
min_delay/hold                          0.00

                                  Weighted
Group (max_delay/setup)     Cost    Weight    Cost    Scenario
-------------------------------------------------------------------------
**default**              0.00     1.00     0.00     m1::c1
**async_default**        0.00     1.00     0.00     m1::c1
**clock_gating_default**  0.00     1.00     0.00     m1::c1
clk                    0.35     1.00     0.35     m1::c1
-------------------------------------------------------------------------
max_delay/setup                         0.35

Constraint                            Cost
---------------------------------------------------
min_delay/hold                         0.00  (MET)
max_delay/setup                        0.35  (VIOLATED)
max_transition                         0.00  (MET)
max_capacitance                        0.00  (MET)
```

```
    min_capacitance                0.00  (MET)
1
```

SEE ALSO

report_timing(2)
report_min_pulse_width(2)
report_min_period(2)

report_corners

Reports corner information.

SYNTAX

string **report_corners**
[-verbose]
[-significant_digits *digits*]
[*corner_names*]

Data Types

digits integer
corner_names list

ARGUMENTS

corner_names

Lists of corners to report. If you do not specify this option, all corners in the current design are reported.

-verbose

Prints additional information about explicit PVT settings on individual cells and ports.

-significant_digits *digits*

Specifies the number of digits after the decimal point to be displayed for time values in the generated report. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** global app option, whose default value is 2. Use this option if you want to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, the tool uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

DESCRIPTION

This command displays information about corners in the current design, including the modes that each corner is activated with.

Multicorner-Multimode Support

This command works on all corners.

SEE ALSO

create_corner(2)
current_corner(2)
report_modes(2)
report_scenarios(2)
shell.common.report_default_significant_digits(3)

report_cross_probing

Reports cross-probing data for specified cells, ports, modules and blocks.

SYNTAX

status **report_cross_probing**
[-nosplit]
[-original_hierarchy]
objects

Data Types

objects collection

ARGUMENTS

-nosplit

Prevents line splitting in the cross-probing report. Most of the design information is listed in fixed-width columns. In the absence of this option, if the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-original_hierarchy

Changes the format of the report by adding an additional line per object to display an object's original RTL hierarchy. Some- times a cell undergoes a series of operations, such as ungroup- ing, name changes, and optimization that collapses the hierarchy and renames the cell. The -original_hierarchy option displays the RTL hierarchical name by which the cell was originally cre- ated and helps to identify where the cell originated.

NOTE : '-original_hierarchy' argument displays original RTL hierarchy only for cell objects and not for port or module/block objects.

objects

Specifies the cell, port, module and block objects to show in the report.

DESCRIPTION

This command displays cross-probing information about specified cells, ports and modules and blocks in a report format.

You must have a design that was analyzed, elaborated and linked or synthesized using this tool. The tool provides information about where the cell or port or module or block was created, specifying which file and line number the object came from. If the object

was tool generated, the file and line number are not applicable and are represented by a hyphen (-). In this case, the report specifies whether the origin of the object comes from UPF, the DFT engine, or another source.

Objects in the design that were merged can result in an object with multiple source locations in the report. Objects with multiple source positions will be shown in the table with multiple filenames, lines and origins separated by comma.

The tool reports the following information for each specified object:

- Type : The type of design object, either a cell or port or module or block.
- Reference : For cells, this is the library reference cell. For ports, this is either IN, OUT, or INOUT. This column will be empty for module & block objects.
- Filename : The base name of the file. If this information is not available, the tool displays a hyphen (-).
- Line : The line number associated with the file. If this information is not available, the tool displays a hyphen (-).
- Origin : The origin associated with the object. For example, this value can be rtl, datapath, upf, clock_gating, self_gating, or dft. If this information is not available, the tool displays a hyphen (-).
- Object : The object name.
- Object / Original Hierarchy : If the -original_hierarchy option is used, you will see this column instead of the Object column.

  For each object, the first row in this column shows the object name.

  For each object, the second row in this column shows the object's original hierarchy if applicable.

At the end of the report, mapping from the base name to the absolute file name is provided. You need to verify the existence of the file because the design or original source files might have been moved or deleted after the design was read in.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows a **report_cross_probing** report:

```
prompt> set objects [get_ports]
{top_i top_o}
prompt> set objects [add_to_collection $objects [get_cells -hierarchical]]
{top_i top_o mid1/bot1/c1 mid1/bot1/c2 mid1/bot2/c1 mid1/bot2/c2 mid1/foo1/c2
mid1/bot1 mid1/bot2 mid1/foo1 mid1}
prompt> report_cross_probing $objects

****************************************
Report : cross_probing
Design : top
Version: ...
Date   : ...
****************************************

Type   Reference     Filename    Line     Origin       Object
--------------------------------------------------------------------------------
port   IN            m.v         2        rtl          top_i
```

```
port   OUT        m.v       3       rtl       top_o
cell   IVA        m.v       21      rtl       mid1/bot1/c1
cell   IVA        m.v       22      rtl       mid1/bot1/c2
cell   IVA        m.v       21      rtl       mid1/bot2/c1
cell   IVA        m.v       22      rtl       mid1/bot2/c2
cell   IVA        m.v       28      rtl       mid1/foo1/c2
cell   bot        m.v       12      rtl       mid1/bot1
cell   bot        m.v       13      rtl       mid1/bot2
cell   myfoo      m.v       14      rtl       mid1/foo1
cell   mid        m.v       4       rtl       mid1

Filename            Full Pathname
--------------------------------------------------------------
m.v                 /usr/joe/design/m.v
```

The following example shows a report with the objects original hierarchy information:

```
prompt> report_cross_probing -original_hierarchy [get_cells U1*]
****************************************
Report  : report_cross_probing
        -original_hierarchy
Design  : data_switch_512
Version : ...
Date    : ...
****************************************
Type  Reference       Filename     Line  Origin  Object/Original Hierarchy
-------------------------------------------------------------------------------
cell  HS65_STF_AOI    test1.v      21536 rtl     U129
                                    I_CORE/I_SUB1
cell  HS65_STF_AND2   test2.v      21548 rtl     U104
                                    I_CORE/I_SUB2
Filename            Full Pathname
-------------------------------------------------------------------------------
test1.v             /usr/joe/rtl/test1.v
test2.v             /usr/joe/rtl/test2.v
```

The following example shows a report with the nosplit option:

```
prompt> report_cross_probing [get_cells U1*] -nosplit
****************************************
Report : report_cross_probing
       -nosplit
Design : top
Version: ....
Date   : ....
****************************************
Type    Reference       Filename        Line     Origin  Object
-------------------------------------------------------------------------------
cell    HS65_STF_AOI    test1_cross_probe_report_with_nosplit_option.v
                                21536    rtl     U129
cell    HS65_STF_AND2   test2.v         21548    rtl     U104

Filename         Full Pathname
-------------------------------------------------------------------------------
test1_cross_probe_report_with_nosplit_option.v
                 /usr/joe/rtl/test1_cross_probe_report_with_nosplit_option.v
test2.v          /usr/joe/rtl/test2.v
```

The following example shows a cross probing report for modules:

```
prompt> report_cross_probing [get_modules]
****************************************
Report : report_cross_probing
Design : top
Version: ....
Date   : ....
****************************************
Type    Reference       Filename        Line     Origin  Object
--------------------------------------------------------------------------------
module              top.v          1       rtl    top
module              top.v          14       rtl    mid
module              top.v          32       rtl    low

Filename         Full Pathname
--------------------------------------------------------------------------------
top.v            /usr/joe/rtl/top.v

1
```

The following example shows a cross probing report for blocks:

```
prompt> report_cross_probing [get_blocks]
****************************************
Report : report_cross_probing
Design : top
Version:
Date   :
****************************************
Type    Reference             FileName              Line    Origin    Object
----------------------------------------------------------------------------------------------------------
block                    top.v              1      rtl      design_lib:top.design

FileName       Full Pathname
----------------------------------------------------------------------------------------------------------
top.v          /usr/joe/rtl/top.v

1
```

SEE ALSO

get_cross_probing_info(2)
cross_probing_filter(2)

report_cross_probing_files

Shows a list of cross-probing files and their status.

SYNTAX

status **report_cross_probing_files**
[-errors_only]

ARGUMENTS

-errors_only

Reports files whose status are not OK.

DESCRIPTION

This command prints out a list of the cross-probing files that are stored in the database and the status of each file. The file status can be one of the following:

Status:

OK - No issues found
Missing - File is missing
No directory permissions - User has no execute permission on the directory in which file exists
No file permissions - User has no read permissions on file
Modified - File contents have been modified

If the status of a file is marked as Modified, the file content has been modified since the file was read. The Missing status might also indicate that the path exists, but it is a directory.

You can use this command to determine if the set of cross-probing file paths stored in the database are valid before using the GUI for cross-referencing. You can also list which RTL files need to be copied to the destination site if the design is being moved to a new location.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of the **report_cross_probing_files** command:

```
prompt> report_cross_probing_files

****************************************
Report : cross_probing_files
Design : top
Version: ....
Date   : ....
****************************************

 Status:
              OK - No issues found
          Missing - File is missing
No directory permissions - User has no execute permission on the directory in which file exists
   No file permissions - User has no read permissions on file
         Modified - File contents have been modified

Filename                    Status
------------------------------------------
/common/rtl/bot.v           OK
/usrsite/joe/designs/mid.v  OK
/usrsite/joe/designs/top.v  OK
```

SEE ALSO

update_cross_probing_files(2)

report_crpr

Reports the clock reconvergence pessimism calculated between specified register clock pins or ports.

SYNTAX

status **report_crpr**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
-from *from_clock_pin*
-to *to_clock_pin*
[-from_clock *from_clock*]
[-to_clock *to_clock*]
[-setup | -hold]
[-significant_digits *digits*]

Data Types

mode_list	list
corner_list	list
scenario_list	list
from_clock_pin	string
to_clock_pin	string
from_clock	string
to_clock	string
digits	integer

ARGUMENTS

-modes *mode_list*

Specifies the modes to be used for reporting. If this option is not given, the command uses the current mode. For each mode, a separate subreport will be printed. It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners to be used for reporting. For each mode being reported, scenarios (if any) of that mode and the corners specified by the **-corners** option will be used for reporting. If this option is not given, the command uses every active scenario of the specified modes. It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios to be used for reporting. Each mode used by the specified scenarios is reported. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the report will use all scenarios of the current mode.

-from ***from_clock_pin***

Specifies the clock pin of the launching sequential device or clock-gating check to be reported. A constrained input port can also be used.

-to ***to_clock_pin***

Specifies the clock pin of the capturing sequential device or clock-gating check to be reported. A constrained output port may also be used.

-from_clock ***from_clock***

Specifies the clock that fans out to the launching sequential device.

-to_clock ***to_clock***

Specifies the clock that fans out to the capturing sequential device.

-setup

Reports the clock reconvergence pessimism for a setup data path. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, **-setup** is assumed.

-hold

Reports the clock reconvergence pessimism for a hold data path. The **-setup** and **-hold** options are mutually exclusive. If neither option is specified, **-setup** is assumed.

-significant_digits ***digits***

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0 through 13. If this option is not specified, the number of significant digits is determined by the **shell.common.report_default_significant_digits** global app option, which has a default value of 2.

DESCRIPTION

This command reports the clock reconvergence pessimism calculated between specified register clock pins or ports. The command displays the following information about the calculation of the clock reconvergence pessimism between the two specified sequential elements:

- The name of the common pin in the clock network
- The clock edges (rising or falling) that propagate through the common point to the launching and capturing registers
- A tabulation of the arrival times for both clock edges at the common point and their associated clock reconvergence pessimism (crp_rise and crp_fall). Separate values are reported for each scenario of the mode.
- The clock reconvergence pessimism used along with the reasoning behind the selection of crp_rise or crp_fall used for that report
- The value of the CRPR threshold.

If the design's **time.remove_clock_reconvergence_pessimism** app option is set to **false**, clock reconvergence pessimism removal is inactive and no report can be generated.

The **-from_clock** and **-to_clock** options allow you to generate a report for only the specified clocks. By default, a report is issued for all clocks that fanout to each sequential device.

Two clock reconvergence pessimism values can be calculated for each potential common point in the design. Only one value is reported, if both values are not needed for timing.

- The crp_rise value is calculated from rising arrival times at the common point.
- The crp_fall value is calculated from falling arrival times at the common point.

The value of clock reconvergence pessimism used in the report depends upon what clock edges propagate through the common point to the clock pins of the launching and capturing devices.

For example, if a rising edge through the common point triggers the launching device and also triggers the capturing device, then a rising clock reconvergence pessimism value (crp_rise) is used. Similar reasoning applies for a falling edge propagating through the common point and subsequently launching and capturing data leading to a falling clock reconvergence pessimism value being used (crp_fall). If a rising edge through the common point launches the data and a falling edge through the common point captures it, then a mismatch in sense occurs.

If a mismatch occurs and the design's **time.clock_reconvergence_pessimism** app option is set to **normal**, then the minimum of crp_rise and crp_fall is used. If a mismatch occurs and the app option is set to **same_transition**, then the clock reconvergence pessimism is reported as 0. This selection process is reported in the selection details section of the report.

When the capturing sequential device is a level-sensitive latch, 2 clock reconvergence pessimism values are reported. The first value corresponds to the opening edge of the latch and appears in the timing report. The second value corresponds to the closing edge of the device. The selection details section also reports the decision-making process behind both clock reconvergence pessimism values. In this case, since the opening and closing clock edges at the latch will always be different for setup, there will always be a mismatch in clock edges for one of them.

With SI enabled, the report prints two types of arrivals for the common point, static and dynamic. Static arrivals exclude dynamic source latency on clocks and SI delta delays. Dynamic arrivals include dynamic latency and SI effects. The CRP calculation uses one of these arrivals and reports in the selection details. For zero-cycle paths with same edge triggering launch and capture, at the common point, dynamic arrivals are used. For all other checks static arrivals are used.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example reports the clock reconvergence pessimism between the sequential elements, F02 and F01, for a setup data path. The report shows details for each mode and the arrival times at the common point for all the corners (scenarios) of the mode. In the following case, it reports only rise arrival times because only rise is used.

```
prompt> report_crpr -from F02/CP -to F01/CP -modes mode1
****************************************
Report : CRP Calculation
      -setup
Design : TOP
Version: <version-number>
Date   : Fri Mar 28 13:21:34 2014
****************************************

  Mode: mode1
  Startpoint: F02/CP (rising edge-triggered flip-flop clocked by CK1)
  Endpoint: F01/CP (rising edge-triggered flip-flop clocked by CK1)
  Common Point: I4/Z
```

```
  Common Clock: CK1
  Launching edge at common point: RISING
  Capturing edge at common point: RISING
  CRPR threshold:    0.00

  Arrival Times        Early    Late     CRP      Corner       Scenario
  ------------------------------------------------------------------------
  Rise                 1.75     1.85     0.10     c1           mode1::c1
  Rise                 1.65     1.80     0.10     c2           mode1::c2
  ------------------------------------------------------------------------

  Selection Details
  ------------------------------------------------------------------------
  Edge Match:     Match, using rise CRP
  ------------------------------------------------------------------------
  clock reconvergence pessimism          0.10     c1           mode1::c1
  clock reconvergence pessimism          0.15     c2           mode1::c2

1
```

The following example reports with level-sensitive latch as endpoint:

```
prompt> report_crpr -from FF2/CP -to LA1/G
****************************************
Report : CRP Calculation
      -setup
Design : TOP
Version: <version-number>
Date   : Fri Mar 28 15:39:05 2014
****************************************

  Mode: default
  Startpoint: FF2/CP (rising edge-triggered flip-flop clocked by net_a)
  Endpoint: LA1/G (positive level-sensitive latch clocked by net_a')
  Common Point: net_a
  Common Clock: net_a
  Launching edge at common point: RISING
  Capturing edge at common point: RISING
  CRPR threshold:    0.00

  Arrival Times        Early    Late     CRP      Corner       Scenario
  ------------------------------------------------------------------------
  Rise                  0.12     0.17     0.05     default      default
  Fall                 0.18     0.21     0.03     default      default
  ------------------------------------------------------------------------

  Selection Details
  ------------------------------------------------------------------------
  Edge Match (opening): Match, using rise CRP
  Edge Match (closing): Mismatch, using min(rise CRP, fall CRP)
  ------------------------------------------------------------------------
  clock reconvergence pessimism (open edge)     0.05   default      default
  clock reconvergence pessimism (close edge)    0.03   default      default

1
```

The following example reports for hold with SI enabled. It shows CRP calculation using higher CRP value using dynamic arrivals.

```
prompt> report_crpr -from ff2/CK -to ff3/CK -hold
****************************************
Report : CRP Calculation
        -hold
Design : basic
Version: <Version>
Date   : Wed Feb 11 15:51:04 2015
****************************************
  Mode: default
  Startpoint: ff2/CK (rising edge-triggered flip-flop clocked by clk)
  Endpoint: ff3/CK (rising edge-triggered flip-flop clocked by clk)
  Common Point: clk
  Common Clock: clk
  Launching edge at common point: RISING
  Capturing edge at common point: RISING
  CRPR threshold:    0.00

  Arrival Times (Static)  Early    Late     CRP       Corner       Scenario
  ------------------------------------------------------------------------------
  Rise                    0.88     1.07     0.19      default      default
  ------------------------------------------------------------------------------

  Arrival Times (Dynamic) Early    Late     CRP       Corner       Scenario
  ------------------------------------------------------------------------------
  Rise                    0.90     1.10     0.20      default      default
  ------------------------------------------------------------------------------

  Selection Details
  ------------------------------------------------------------------------------
  Arrival Times:   Using Dynamic Arrivals
  Edge Match:      Match, using rise CRP
  ------------------------------------------------------------------------------
  clock reconvergence pessimism             0.20      default      default

1
```

The following example reports for setup with SI enabled. Static arrivals are used and smaller CRP value selected for setup.

```
prompt> report_crpr -from ff2/CK -to ff3/CK -setup
****************************************
Report : CRP Calculation
        -setup
Design : basic
Version: <version>
Date   : Wed Feb 11 15:51:04 2015
****************************************
  Mode: default
  Startpoint: ff2/CK (rising edge-triggered flip-flop clocked by clk)
  Endpoint: ff3/CK (rising edge-triggered flip-flop clocked by clk)
  Common Point: clk
  Common Clock: clk
  Launching edge at common point: RISING
  Capturing edge at common point: RISING
  CRPR threshold:    0.00

  Arrival Times (Static)  Early    Late     CRP       Corner       Scenario
  ------------------------------------------------------------------------------
```

```
    Rise                0.88    1.07    0.19      default     default
    ---------------------------------------------------------------------------

    Arrival Times (Dynamic) Early    Late    CRP       Corner      Scenario
    ---------------------------------------------------------------------------
    Rise                0.90    1.10    0.20      default     default
    ---------------------------------------------------------------------------

    Selection Details
    ---------------------------------------------------------------------------
    Arrival Times:   Using Static Arrivals
    Edge Match:      Match, using rise CRP
    ---------------------------------------------------------------------------
    clock reconvergence pessimism            0.19      default     default

  1
```

SEE ALSO

report_timing(2)
shell.common.report_default_significant_digits(3)
time.remove_clock_reconvergence_pessimism(3)
time.clock_reconvergence_pessimism(3)

report_datapath_architecture_options

Displays datapath strategies available to the current design.

SYNTAX

status **report_datapath_architecture_options**
[**-nosplit**]

ARGUMENTS

-nosplit

Does not split lines if column overflows.

DESCRIPTION

This command lists the customized datapath strategies available to the current design. These strategies provide control over some of the optimizations performed by datapath generators. You can use the **set_datapath_architecture_options** command to set the strategies on specific modules, specific cells, or the whole design.

EXAMPLES

The following is an example of a report generated by **report_datapath_architecture_options**:

```
prompt> report_datapath_architecture_options
****************************************
Report : report_datapath_architecture_options
Design : top
Version: 2.1.5
Date   : Sat Sep 24 06:28:55 2016
****************************************

================================================================================
| Object      | Type     | Datapath Generation Options                       |
================================================================================
| top         | design   | invOutAdderCell=true optimizeFor=area             |
| middle      | module   | boothEncoding=false optimizeFor=speed             |
| mult_15     | cell     | boothEncoding=false multRadix4=false multNandBased=true |
```

==========

SEE ALSO

set_datapath_architecture_options(2)

report_delay_calculation

Displays the actual calculation of a timing arc delay value for cell, net, or library cell timing arcs.

SYNTAX

status **report_delay_calculation**
[-from *pin_port_list*]
[-to *pin_port_list*]
[-mode *mode*]
[-corner *corner*]
[-scenario *scenario*]
[-transition *time*]
[-load *capacitance*]
[-voltage *voltage*]
[-temperature *temperature*]
[-clock *clock*]
[-source_rise]
[-source_fall]
[-pvt]
[-significant_digits *digits*]
[-process_number *number*]
[-process_label *process*]
[-min]
[-crosstalk]
[-derate]
[*arc_list*]

Data Types

pin_port_list	collection
mode	collection
corner	collection
scenario	collection
time	float
capacitance	float
voltage	float
temperature	float
clock	collection
digits	integer
number	float
process	string
arc_list	collection

ARGUMENTS

-from *pin_port_list*

Specifies the starting point of timing arcs within the block. You cannot specify a hierarchical pin.

This option must be used with the **-to** option and is mutually exclusive with the *arc_list* argument.

-to *pin_port_list*

Specifies the ending point of timing arcs within the block. You cannot specify a hierarchical pin.

This option must be used with the **-from** option and is mutually exclusive with the *arc_list* argument.

-mode *mode*

Specifies the mode to use for reporting.

The command uses the scenario (if any) of the specified mode and the corner specified by the **-corner** option for reporting.

If you do not specify any of the **-mode**, **-corner**, or **-scenario** options, the command uses data from the current scenario to generate the report.

It is an error to specify this option with the **-scenario** option.

-corner *corner*

Specifies the corner to use for reporting.

The command uses the scenario (if any) of the specified corner and the mode specified by the **-mode** option for reporting.

If you do not specify any of the **-mode**, **-corner**, or **-scenario** options, the command uses data from the current scenario to generate the report.

It is an error to specify this option with the **-scenario** option.

-scenario *scenario*

Specifies the scenario to use for reporting.

If you do not specify any of the **-mode**, **-corner**, or **-scenario** options, the command uses data from the current scenario to generate the report.

It is an error to specify this option with the **-mode** or the **-corner** options.

-transition *time*

Specifies an override for the rising and falling input transition value used for delay calculations. The specified value is in main library units.

For Arnoldi calculation, the specified input transition is always applied to the input pin of the driving cell. For Elmore calculation, the specified input transition is applied to the from pin of the given arc.

If you do not specify this option, the transition time calculated by the timer is used for cell and net arcs and zero is used for library cell arcs.

-load *capacitance*

Specifies an override for the load capacitance of the arcs. The specified value is in main library units.

This option allows you to analyze the behavior of a cell with different values of load capacitance.

If you do not specify this option, the timer-calculated load capacitance is used for cell and net arcs and zero is used for library cell arcs.

This option does not work with Arnoldi delay calculation.

-voltage *voltage*

Specifies an override for the supply voltage used to calculate the delay of a cell or library cell arc. The specified value is in main library units.

This option allows you to analyze the behavior of a cell with different values of supply voltage. For this option to be effective, the cell's library data must contain characterization data for multiple values of supply voltage.

If you do not specify this option, the block's top-level primary voltage is used for library cell arcs.

-temperature *temperature*

Specifies an override for the temperature used to calculate the delay of a cell or library cell arc. The specified value is in main library units.

This option allows you to analyze the behavior of a cell with different values of temperature. For this option to be effective, the cell's library data must contain characterization data for multiple temperatures.

If you do not specify this option, the block's top-level temperature is used for library cell arcs.

-clock *clock*

Uses existing transition data for the specified clock and clock timing derate values (if any) to calculate delays.

If the cell or net arcs being reported are not part of the clock's network, the command issues a warning message. If the clock does not belong to the mode being reported, the command tries to use a clock of the same name from the correct mode.

If you do not specify this option, the command uses data transitions and derate values.

When you specify this option, you must also use either the **-source_rise** or **-source_fall** option.

-source_rise

Uses the rising clock source edge to determine the transition time at the source node of the timing arc.

This option is valid only when you specify the **-clock** option.

-source_fall

Uses the falling clock source edge to determine the transition time at the source node of the timing arc.

This option is valid only when you specify the **-clock** option.

-pvt

Reports details of the specified and effective PVT parameters of the from- and to-pins.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 through 13. The default is 2.

This option overrides the value set by the **shell.common.report_default_significant_digits** application option.

-process_number *number*

Overrides the operating process number.

-process_label *process*

Attaches a process label to the libraries that are read.

By default, the process label is the name of the default operating condition. Process labels are a means of grouping libraries together for scaling.

-min

Reports the minimum delay values instead of maximum values if the report does not print both.

-crosstalk

Reports details related to crosstalk calculation.

You should not use this option with override options, such as **-voltage** and **-temperature**, that change uncoupled delays. This option reports crosstalk calculation results based only on the actual uncoupled delay shown in the **report_timing** report.

-derate

Reports derating components and derated delay.

arc_list

Specifies the cell, net, or library cell timing arcs to report. This option cannot be used with the **-from** and **-to** options.

DESCRIPTION

This command provides detailed timing calculation information about the specified cell, net, or library cell timing arcs. You can use this information to debug or verify timing data in a reference library. For cells and library cells, you can report on both check and delay arcs.

Operating conditions (process, voltage, and temperature values) are considered when making delay calculations. The calculations are based on the PVT and timing data of the specified scenario. For library cell arcs, the command uses the block's top-level PVT parameters.

If you specify pin/port lists with the **-from** and **-to** options, the command reports all cell or net arcs between any of the **-from** and **-to** pins/ports. If the same input pin is specified as both the **-from** and **-to** pin, the command reports the pin's driving_cell (or **set_drive**) calculations. You can specify a collection of cell, net, or library cell timing arcs instead of the **-from** and **-to** options. To create a collection of timing arcs, use the **get_timing_arcs** command.

If you specify the **-clock** option, the command uses the clock transition data and uses timing derate values (if any) for the calculations. If you do not specify this option, the command uses the data transitions and derate values. If the cell or net arcs being reported are not part of the specified clock's network, the command issues a warning message. The command also issues a warning message if you use the **-clock** option with library cell arcs or check arcs (which are never part of a clock network). If you specify the **-clock** option, you must also specify either the **-source_rise** or **-source_fall** option. This option specifies which clock source edge to use to determine the transition time at the source node of the timing arc.

You can use the **-transition**, **-load**, **-voltage**, and **-temperature** options to override the values normally used by the timer. This allows you to investigate how a cell's behavior is affected by changes to these parameters.

For Elmore delay calculation, the report includes both early and late calculations. For Arnoldi delay calculation, the report includes either early or late calculation based on the **-min** option.

The generated timing report includes values for stored delay and output slew in addition to the delay and output slew values. The stored delay and output slew values are the values that would be seen in a timing report through the specified stage. If a timing arc has annotations or derating factors applied to it, its stored delay value might differ from its delay value. If multiple timing arcs arrive at an output pin, the stored output slew value is the slowest transition to the output pin. A value of "..." in the stored delay or stored output slew column indicates that there is no stored data because setup, hold, or maximum transition is disabled for the scenario for which the stage is reported.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenario** option or the **-corner** and **-mode** options.

EXAMPLES

The following example shows a cell arc's derating information with **-derate** option when it honors side file table. The "POCVM coefficient" indicates that it is using side file table for POCV annotation. "Cell delay" is the delay before applying derates, and "Cell delay derated" is the delay after applying derates.

```
prompt> report_delay_calculation -from U5/A -to U5/Z -sig 4 -derate
                          Rise        Fall
---------------------------------------------------------------------------
Cell delay                = 0.0492      0.0497
POCVM coefficient         = 0.1000      0.1000
POCVM coef scale factor   = 1.0000      1.0000
POCVM distance derate     = 1.0000      1.0000
POCVM guardband           = 1.0000      1.0000
Incremental derate        = 0.0000      0.0000
Cell delay derated        = 0.0492      0.0497
Cell delay sigma          = 0.0049      0.0050

Cell delay derated = "Cell delay" *
( "POCVM guardband" * "POCVM distance derate" + "Incremental derate" )
Cell delay sigma = "Cell delay" *
( "POCVM guardband" * "POCVM coefficient" * "POCVM coef scale factor" )
```

The following example shows a cell arc's derating information with **-derate** option when it honors LVF library data. The "POCVM delay sigma" indicates that it is using LVF library for POCV annotation.

```
prompt> report_delay_calculation -from U5/A -to U5/Z -sig 4 -derate
                          Rise        Fall
---------------------------------------------------------------------------
Cell delay                = 0.0492      0.0497
POCVM delay sigma         = 0.0492      0.0497
POCVM coef scale factor   = 1.1000      1.1500
POCVM distance derate     = 1.0000      1.0000
POCVM guardband           = 1.2000      1.2000
Incremental derate        = 0.1000      0.0500
Cell delay derated        = 0.0640      0.0621
Cell delay sigma          = 0.0650      0.0685

Cell delay derated = "Cell delay" *
( "POCVM guardband" * "POCVM distance derate" + "Incremental derate" )
Cell delay sigma = "POCVM delay sigma" *
( "POCVM guardband" * "POCVM coef scale factor" )
```

The following example shows a cell arc's derating information with **-derate** option when it honors LVF library data and slew variation is on. "POCVM delay sigma" and "POCVM output transition sigma" are the intrinsic delay sigma and intrinsic transition sigma. Slew induced variation includes "POCVM delay sigma induced by input transition variation" and "POCVM output transition sigma induced by input transition variation". "POCVM combined delay sigma" is the combination of "POCVM delay sigma" and "POCVM delay sigma induced by input transition variation", which is the final delay sigma before applying derates. "POCVM combined output

transition sigma" has the similar rule.

```
prompt> report_delay_calculation -from U5/A -to U5/Z -sig 4 -derate
                          Rise        Fall
---------------------------------------------------------------------------
Cell delay                = 0.0492      0.0497
POCVM delay sigma         = 0.0492      0.0497
POCVM delay sigma induced by input transition variation = 0.0055 0.0099
POCVM combined delay sigma      = 0.0556      0.0576
POCVM output transition sigma   = 0.0423      0.0300
POCVM output transition sigma induced by input transition variation = 0.0026 0.0027
POCVM combined output transition sigma = 0.0424   0.0301
POCVM input transition sigma    = 0.0928      0.0614
POCVM coef scale factor         = 1.1000      1.1500
POCVM distance derate           = 1.0000      1.0000
POCVM guardband           = 1.2000      1.2000
Incremental derate        = 0.1000      0.0500
Cell delay derated        = 0.0640      0.0621
Cell delay sigma          = 0.0733      0.0795

Cell delay derated = "Cell delay" *
( "POCVM guardband" * "POCVM distance derate" + "Incremental derate" )
Cell delay sigma = "POCVM combined delay sigma" *
( "POCVM guardband" * "POCVM coef scale factor" )
```

The following example shows a cell arc's derate information with **-derate** option when it honors regular derates.

```
prompt> report_delay_calculation -from U5/A -to U5/Z -sig 4 -derate
                          Rise        Fall
---------------------------------------------------------------------------
Cell delay                = 0.0492      0.0497
Cell derate               = 1.2000      1.2000
Incremental derate        = 0.1000      0.0500
Cell delay derated        = 0.0640      0.0621

Cell delay derated = "Cell delay" * ( "Cell derate" + "Incremental derate" )
```

The following example shows a net arc's derate information with **-derate** option when it honors regular derate and SI analysis is enabled.

```
prompt> report_delay_calculation -from BlkA/U7/Z -to U5/A -derate -sig 4
                          Rise        Fall
---------------------------------------------------------------------------
Net delay                 = 0.0068      0.0068
Net derate                = 1.0000      1.0000
Incremental derate        = 0.1000      0.0500
Net delay derated         = 0.0075      0.0071

Net delay derated = "Net delay" * ( "Net derate" + "Incremental derate" )

Dynamic derate            = 1.1000      1.0500
Delta delay               = 0.0260      0.0076
Delta delay derated       = 0.0286      0.0080

Delta delay derated = "Dynamic derate" * "Delta delay"
```

The following example shows a net arc's derate information with **-derate** option when it honors side file. Note that net arc only

honors side file for POCV annotation.

```
prompt> report_delay_calculation -from BlkA/U7/Z -to U5/A -derate -sig 4
                          Rise         Fall
---------------------------------------------------------------------------
Net delay                 = 0.0068       0.0068
POCVM coefficient         = 0.1000       0.1000
POCVM coef scale factor   = 1.1000       1.1500
POCVM distance derate     = 1.0000       1.0000
POCVM guardband           = 1.2000       1.2000
Incremental derate        = 0.1000       0.0500
Net delay derated         = 0.0088       0.0085
Net delay sigma           = 0.0009       0.0009

Net delay derated = "Net delay" *
( "POCVM guardband" * "POCVM distance derate" + "Incremental derate" )
Net delay sigma   = "Net delay" *
( "POCVM guardband" * "POCVM coefficient" * "POCVM coef scale factor" )
```

SEE ALSO

get_timing_arcs(2)
report_stage(2)
report_timing(2)
report_lib(2)

report_density_gradient_options

Reports information for the **report_placement -hard_macro_density_gradient_violations** command.

SYNTAX

status **report_density_gradient_options**

DESCRIPTION

This command reports the options set by the **set_density_gradient_options** command. If an option is not specified, the default value is reported. Both the **set_density_gradient_options** and **report_density_gradient_options** commands support the **report_placement -hard_macro_density_gradient_violations** command.

EXAMPLES

The following example sets density gradient options with the **set_density_gradient_options** command and reports the settings with the **report_density_gradient_options** command.

```
prompt> set_density_gradient_options -white_space_density {M1 0 M2 0.6} \
      -edge_zone_width {M1 50 M2 60} \
      -outer_zone_width {M1 50 M2 60} \
      -gradient_tolerance {M1 0.3 M2 0.5} \
      -min_macro_size 600 \
      -cluster_threshold 4

prompt> report_density_gradient_options
****************************************
Report : report_density_gradient_options
 ...

****************************************
------------------------------------------------------------
White Space Density Report:
      Layer                  White space density
      ------------------------------------------------------
      M1                     0%
      M2                     60%
      Others                  35%
------------------------------------------------------------
Edge Zone Width Report:
```

```
    Layer                    Edge zone width(um)
    ------------------------------------------------------------
    M1                       50
    M2                       60
    Others                    150
------------------------------------------------------------
Outer Zone Width Report:
    Layer                    Outer zone width(um)
    ------------------------------------------------------------
    M1                       50
    M2                       60
    Others                    150
------------------------------------------------------------
Gradient Tolerance Report:
    Layer                    Gradient tolerance value
    ------------------------------------------------------------
    M1                       30%
    M2                       50%
    Others                    30%
------------------------------------------------------------
Min macro size(um):                600
------------------------------------------------------------
Cluster Threshold(um):              4
------------------------------------------------------------
1
```

SEE ALSO

set_density_gradient_options(2)
report_placement(2)

report_design

Reports netlist, floorplan, routing, and library information for the current block. By default, the command reports the number of standard cells, macro cells, and other elements in the design.

SYNTAX

status **report_design**
[-library]
[-netlist]
[-floorplan]
[-routing]
[-all]
[-nosplit]
[-hierarchical]

ARGUMENTS

-library

Reports information about the reference libraries for the current block. The report includes the search path, units, technology file, number of scenarios in the design, standard cell information and statistics, and some details of the reference libraries.

The report is divided into sections for search path, units, scenarios, and library details. The search path section prints value of the **search_path** variable.

The units section prints the units from the main reference library. Note that the command does not consider user-defined units.

The technology file section prints the technology file that was used to build the current block.

The scenario section reports the number of active and inactive scenarios. If the block has only one scenario, the number of active scenarios is 1 and number of inactive scenarios is 0.

The next section reports information about the standard cells in tall reference libraries. The command considers only those library cells that have a timing view and whose **design_type** attribute is set to **lib_cell**.

The library details section reports details for each standard cell library. The command prints the full path to each library, along with the list of db files that were used to build this library and the full path to these logic library (.db) files.

The **-hierarchical** option has no effect on this option.

-netlist

Reports information about the library cells and the number of instances in the netlist. The report includes the number of cell instances, library cells, flat nets, ports, and pins. When you use this option with the **-all** option, the command also generates a netlist report.

The report is divided into sections for cell instances, reference designs, nets, fanout, and port and pin information. In the cell instance section, the report includes the gate count, percentage of total leaf cells and of total cell area, and the projected cell area in both square microns and sites. If a cell instance does not have a site definition, the command calculates site area based on a cell instance with minimum site height.

The report also reports cells with various design types (as determined by the **design_type** attribute); cells with special types such as level shifter, isolation, switch, always on, retention, and tie off; high, low, and normal threshold voltage cells; netlist and physical-only cells; fixed and movable cells; combinational and sequential cells; buffers and inverters; spare cells; integrated clock-gating (ICG) cells; flip-flops; latches; antenna cells; multiplexer logic; and multiheight cells. The calculations are based on the height of the cell with respect to the site height of the cell instance.

In the reference design section, the command reports the design type, instance count, width, height, area, pin density, site name, and site area for each library cell. Pin density is the ratio of the pins and the area of the library cell. Site name and site area are reported if the library cell has a site definition.

In the flat net section, the command reports the count, floating nets, total vias, and the ratio of net counts to total cell counts. The report also contains pin count and fanout statistics for different ranges of fanout and pin count. A net is considered floating when the pin count is less than or equal to one or when the pin count is same as the fanout and there are no inout pins.

In the leaf pins and ports section, the command reports the counts of power and ground pins and ports. The counts are based on the direction of the pin or port.

To report hierarchical netlist information, use the **-hierarchical** option with this option.

-floorplan

Displays the floorplan information for the design.

The floorplan information includes the following sections:

- Core and chip area : Reports information about the core area and chip area.
- Site row : Reports the name, width, height, number of rows, number of tiles, and area for each site definition.
- Blockage : Reports the name, count, and area for each blockage type.
- Power domain : Reports the name, voltage area name, primary power net, and primary ground net for each power domain.
- Voltage area : Reports the name, number of shapes, area, target utilization, and bounding box coordinates for each voltage area.
- Group bound : Reports the number of group bounds in the design.
- Exclusive move bound : Reports the name, area, utilization, and bounding box coordinates for each exclusive move bound in the design.
- Hard and soft move bound: Reports the name, area, utilization, bounding box coordinates, and type for each move bound in the design.
- Routing guide : Reports the type, count, and area for each routing guide in the design.
- Multibit register : Reports the count of different multibit registers in the design and the multibit banking ratio.
- Relative placement group : Reports the number of top-level relative placement (RP) groups in the design, the total number of RP cells, and the RP cell area as a percentage of total leaf cell area of design.
- Layer : Reports the details of layers with known routing directions. For each layer, the command reports the layer name, routing direction, whether the layer is ignored, pitch, default width, minimum width, minimum spacing, and same net minimum spacing. If there is a routing direction override for a specific layer, the command reports the override direction.

To report hierarchical floorplan information, use the **-hierarchical** option with this option.

-routing

Reports information and statistics related to routing. The report includes signal wiring, clock wiring and PG wiring statistics, as well as via instances and the double via conversion rates. The command reports the details of the final routing state.

By default, the command reports the routing information only for the top-level of the physical hierarchy. To aggregate the information across all levels of the physical hierarchy, use the **-hierarchical** option.

The summary section reports the total number of all wires across all layers and their total length. It also reports the total number of contacts (vias) across all the layers, including simple vias, simple array vias, and custom vias.

In the final wiring statistics table, the command reports the following information for each layer: the total number of wires on the layer, its percentage with respect to the total wire count for the layer, the total length of wires on the layer, and its percentage with respect to the total wire length.

In the PG wiring section, for each layer, the command reports the power and ground wiring statistics. In this section, the command reports only those wires whose **net_type** attribute is set to **power** or **ground**.

In the shape pattern wiring statistics table, the command reports the shape pattern wiring statistics for each layer. This command only reports the path pattern type in this wiring statistics table.

In the horizontal and vertical wire distribution section, the command reports the total length of horizontal wires or horizontal segments of wires and its percentage with respect to the total horizontal wire length. Vertical routing distribution is reported similarly.

Except for PG wiring statistics, all other wiring statistics in this report consider only signal wires.

The final via statistics section reports via information for each via layer. All simple via definitions and simple array via definitions associated with a particular via layer are listed. The report also includes the number of instances of that via definition and the percentage of that number with respect to the total number of vias on that via layer.

The character string "(rot)" is appended to the via definition name for rotated instances of simple via definitions. A rotated via is a via with an orientation other than R0, R180, MX, or MY. The character string "mXn" is appended to the via definition name for simple via arrays, which denotes a via array with m rows and n columns. All such vias are counted as "double vias". For simple vias (one row and one column), the via definition name is reported without a suffix. The double via conversion rate is reported for each via layer. The rate is the ratio of the total number of instances of "double vias" (as explained above) to the total number of all vias in this layer, expressed as a percentage. Note that this calculation considers only double vias; other types of optimized vias are not considered.

The overall double via conversion rate is the ratio of the total number of double via instances to the total number of vias, expressed as a percentage. The command reports the conversion rate both considering all vias (both PG and detail routing) and considering only detail routing vias. If the design contains custom vias, the command reports them separately in the custom via statistics section. For custom vias, the command reports only the via definition name and the total number of instances of this custom via. Custom vias might have more than one cut layers associated with the via, so this information is not classified by layer.

DRC summary information is also reported if this information is stored in the design when the routing is done.

To report hierarchical routing information, use the **-hierarchical** option with this option.

-all

Reports floorplan, netlist, routing, and library information. The output of the **report_design -all** command is equivalent to running **report_design -floorplan -netlist -routing -library**.

-nosplit

Retains long lines, even if the lines are longer than 80 columns. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing scripts to extract information from the report output.

-hierarchical

Includes all objects of the physical hierarchies in the report. This option applies to reports generated by the **-netlist**, **-floorplan**, and **-routing** options, as well as the default summary.

DESCRIPTION

This command lists information and statistics about various netlist, floorplan, library, and routing objects in the current block. When you run the command with no options, the command performs a statistical analysis of various aspects of the design, such as cells, nets, and area, and generates a table with the results.

This command supports hierarchical reporting, which is enabled by the **-hierarchical** option. When you specify this option, reporting includes the objects of the physical hierarchies beneath the top level.

EXAMPLES

The **report_design** command displays the following information when no options are specified.

```
prompt> report_design
****************************************
Report : design
Design : my_design
****************************************

Total number of std cells in library : 536
Total number of dont_use lib cells   : 22
Total number of dont_touch lib cells : 22
Total number of buffers              : 26
Total number of inverters            : 13
Total number of flip-flops           : 120
Total number of latches              : 60
Total number of ICGs                 : 20

Cell Instance Type  Count       Area
---------------------------------------
TOTAL LEAF CELLS    35211  3869717.457
Standard cells      34620   716005.644
Hard macro cells       41   557965.758
Soft macro cells        0        0.000
Always on cells         0        0.000
Physical only         478  1102905.254
Fixed cells             0        0.000
Moveable cells      35211  3869717.457
Sequential           6052   787596.632
Buffer/inverter      2086   591885.360
ICG cells               0        0.000

Logic Hierarchies                 : 34
Design Masters count              : 34
Total Flat nets count             : 44572
Total FloatingNets count          : 44572
Total no of Ports                 : 34
```

```
Number of Master Clocks in design    : 0
Number of Generated Clocks in design : 0
Number of Path Groups in design      : 6 (0 of them Non Default)
Number of Scan Chains in design      : 0
List of Modes                  : default
List of Corners                : default
List of Scenarios              : default

Core Area                      : 2051583.174
Chip Area                      : 4943796.228
Total Site Row Area            : 2051583.174
Number of Blockages            : 0
Total area of Blockages        : 0.000
Number of Power Domains        : 1
Number of Voltage Areas        : 1
Number of Group Bounds         : 0
Number of Exclusive MoveBounds      : 4
Number of Hard or Soft MoveBounds   : 0
Number of Multibit Registers        : 0
Number of Multibit LS/ISO Cells     : 0
Number of Top Level RP Groups       : 0
Number of Tech Layers          : 6

Total wire length              : 0.00 micron
Total number of wires          : 0
Total number of contacts       : 0
1
```

The following example reports library information for the current block.

```
prompt> report_design -library
****************************************
Report : design
       -library
Design : my_design
****************************************
--------------------------------------------------------------------------------
                        LIBRARY INFORMATION
--------------------------------------------------------------------------------

Search path : /u/libs/my_design

Units :
   time            : 1.00ns
   resistance      : 1.00kOhm
   capacitance     : 1.00pF
   voltage         : 1.00V
   current         : 1.00uA
   power           : 1.00pW

Tech file : /u/ref_lib/my_design/src/cb13_6m.tf

Number of active scenarios     = 1
Number of inactive scenarios   = 0

TOTAL NUMBER OF STANDARD CELLS       = 536
```

```
Total number of dont_use lib cells     = 22
Total number of dont_touch lib cells    = 22

Total number of buffers        = 26
Total number of inverters       = 13
Total number of flip-flops     = 120
Total number of latches         = 60
Total number of ICGs           = 20


Library : sc
  File path : /u/libs/my_design/sc.ndm
  Source .db libs :
    cb13fs120_tsmc_max : /u/ref_lib/my_design/src/sc_max.db
    cb13fs120_tsmc_min : /u/ref_lib/my_design/src/sc_min.db

Library : io
  File path : /u/libs/my_design/io.ndm
  Source .db libs :
    cb13io320_tsmc_max : /u/ref_lib/my_design/src/io_max.db
    cb13io320_tsmc_min : /u/ref_lib/my_design/src/io_min.db

Library : special
  File path : /u/libs/my_design/special.ndm
  Source .db libs :
    cb13special_max : /u/ref_lib/my_design/src/special_max.db
    cb13special_min : /u/ref_lib/my_design/src/special_min.db

Library : ram4x32
  File path : /u/libs/my_design/ram4x32.ndm
  Source .db libs :
    ram4x32_max : /u/ref_lib/my_design/src/ram4x32_max.db
    ram4x32_min : /u/ref_lib/my_design/src/ram4x32_min.db

Library : ram8x64
  File path : /u/libs/my_design/ram8x64.ndm
  Source .db libs :
    ram8x64_max : /u/ref_lib/my_design/src/ram8x64_max.db
    ram8x64_min : /u/ref_lib/my_design/src/ram8x64_min.db

Library : ram32x64
  File path : /u/libs/my_design/ram32x64.ndm
  Source .db libs :
    ram32x64_max : /u/ref_lib/my_design/src/ram32x64_max.db
    ram32x64_min : /u/ref_lib/my_design/src/ram32x64_min.db

Library : ram16x128
  File path : /u/libs/my_design/ram16x128.ndm
  Source .db libs :
    ram16x128_max : /u/ref_lib/my_design/src/ram16x128_max.db
    ram16x128_min : /u/ref_lib/my_design/src/ram16x128_min.db
1
```

The following example reports netlist information for the current block.

```
prompt> report_design -netlist
****************************************
```

```
Report : design
       -netlist
Design : my_design
****************************************

--------------------------------------------------------------------------------
                     NETLIST INFORMATION
--------------------------------------------------------------------------------

CELL INSTANCE INFORMATION
--------------------------------------------------------------------------------
Cell Instance Type         Count % of        Area  % of siteAreaPerSite
                           total             total
--------------------------------------------------------------------------------
TOTAL LEAF CELLS           35211  100  3869717.457   100 unit:2616851

  Standard cells           34620   98   716005.644    18 unit:473267
  Filler cells                 0    0        0.000     0
  Diode cells                  0    0        0.000     0
  Module cells                 0    0        0.000     0
    Soft macro cells           0    0        0.000     0
  Hard macro cells            41    0   557965.758    14 unit:378076
  Abstract cells               0    0        0.000     0
  Black box cells              0    0        0.000     0
  Analog block cells           0    0        0.000     0
  Pad cells                  550    1  2595746.054    67 unit:1765508
    Flip-chip pad cells        0    0        0.000     0
  Cover cells                  0    0        0.000     0
  Flip-chip driver cells       0    0        0.000     0
  Corner pad cells             0    0        0.000     0
  Pad spacer cells             0    0        0.000     0
  Others                       0    0        0.000     0

Special cells                  0    0        0.000     0
  Level shifter                0    0        0.000     0
  Isolation                    0    0        0.000     0
  Switch                       0    0        0.000     0
  Always on                    0    0        0.000     0
  Retention                    0    0        0.000     0
  Tie off                      0    0        0.000     0

LVT                            0    0        0.000     0
HVT                            0    0        0.000     0
Normal VT                      0    0        0.000     0
Others                     35211  100  3869717.457   100 unit:2616851

Netlist cells              34733   98  2766812.203    71 unit:1846239
Physical only                478    1  1102905.254    28 unit:770612

Fixed cells                    0    0        0.000     0
Moveable cells             35211  100  3869717.457   100 unit:2616851

Combinational              28697   81  2310957.971    59 unit:1538630
Sequential                  6052   17   787596.632    20 unit:528697
Others                       462    1   771162.854    19 unit:549524
```

```
Buffer                  66    0   580907.757    15 unit:387141
Inverter              1981    5     9679.534     0 unit:6398
Buffer/inverter       2086    5   591885.360    15 unit:394397

Spare cells            478    1  1102905.254    28 unit:770612
ICG cells                0    0        0.000     0
Flip-flop cells       6046   17   787421.136    20 unit:528581
Latch cells              6    0      175.496     0 unit:116
Antenna cells            0    0        0.000     0

Mux logic              480    1     5927.542     0 unit:3918
Dont touch cells        41    0   557965.758    14 unit:378076
Size only cells          3    0       42.361     0 unit:28

Double height            0    0        0.000     0
Triple height            0    0        0.000     0
More than triple height      550    1  2595746.054    67 unit:1765508

Logic Hierarchies       : 34

REFERENCE DESIGN INFORMATION
--------------------------------------------------------------------------------------------
Number of reference designs used:127
--------------------------------------------------------------------------------------------
Name         Type        Count    Width    Height       Area   PinDens SiteName siteArea
--------------------------------------------------------------------------------------------
nr02d0       lib_cell    10634     1.64     3.69      6.052    0.826 unit         4
ad01d0       lib_cell     7840     8.20     3.69     30.258    0.231 unit        20
sdcrq1       lib_cell     2468    12.71     3.69     46.900    0.171 unit        31
inv0d1       lib_cell     1915     1.23     3.69      4.539    0.881 unit         3
sdcfq1       lib_cell     1616    12.71     3.69     46.900    0.171 unit        31
nd02d1       lib_cell     1575     1.64     3.69      6.052    0.826 unit         4
xr03d1       lib_cell     1462     6.97     3.69     25.719    0.233 unit        17
sdcrn1       lib_cell      942    12.71     3.69     46.900    0.171 unit        31
ah01d0       lib_cell      906     4.92     3.69     18.155    0.330 unit        12
sdnrq1       lib_cell      594    12.30     3.69     45.387    0.154 unit        30
xr02d1       lib_cell      577     4.51     3.69     16.642    0.300 unit        11
oai21d1      lib_cell      468     3.69     3.69     13.616    0.441 unit         9
invbdk       lib_cell        1     9.02     3.69     33.284    0.120 unit        22
ora211d1     lib_cell        1     3.69     3.69     13.616    0.514 unit         9
or02d0       lib_cell        1     2.05     3.69      7.565    0.661 unit         5
aoi31d1      lib_cell        1     4.10     3.69     15.129    0.463 unit        10
sdcrb2       lib_cell        1    13.94     3.69     51.439    0.175 unit        34
nd02d2       lib_cell        1     2.87     3.69     10.590    0.472 unit         7
nd12d1       lib_cell        1     2.46     3.69      9.077    0.551 unit         6
oai31d1      lib_cell        1     4.51     3.69     16.642    0.421 unit        11
CLKMUL       macro           1   120.00   100.00  12000.000    0.000
nr04d1       lib_cell        1     4.10     3.69     15.129    0.463 unit        10
lanhq1       lib_cell        1     5.74     3.69     21.181    0.236 unit        14
nr13d2       lib_cell        1     4.92     3.69     18.155    0.330 unit        12
aoim21d1     lib_cell        1     4.10     3.69     15.129    0.397 unit        10
sdcrn4       lib_cell        1    13.53     3.69     49.926    0.160 unit        33
invbd7       lib_cell        1     3.69     3.69     13.616    0.294 unit         9

NET INFORMATION
-----------------------------------------------------------------
```

```
NetType             Count FloatingNets       Vias Nets/Cells
-------------------------------------------------------------------
Total               44572        658          0     1.266
Signal              44536        628          0     1.265
Power                   3          0          0     0.000
Ground                  3          0          0     0.000
Analog Signal           0          0          0     0.000
Analog Ground           0          0          0     0.000
Analog Power            0          0          0     0.000
Clock                   0          0          0     0.000
Tie Low                22         22          0     0.001
Tie High                8          8          0     0.000
Others                  0          0          0     0.000

NET FANOUT AND PIN COUNT INFORMATION
-----------------------------------------------------
Fanout      Netcount    netPinCount    NetCount
-----------------------------------------------------
<2           33843    <2              639
2             6864    2             33280
3             1049    3              6752
4              361    4              1084
5              162    5               359
6-10          1057    6-10           1204
11-20         1170    11-20          1182
21-30            7    21-30             9
31-50           32    31-50            32
51-100          12    51-100           16
101-500          3    101-500           3
501-1000         5    501-1000          5
>1000            7    >1000             7

PORT AND PIN INFORMATION
-------------------------------------------------------------------------------
Type       Total    Input   Output    Inout    3-st    Power   Ground
-------------------------------------------------------------------------------
Total     213825   168588    44849       44     684    34937    34937
Macro       2541     1463      646        0     640       41       41
Ports         74       57       63       46       0        1        1
-------------------------------------------------------------------------------
1
```

The following example reports floorplan information for the current block.

```
prompt> report_design -floorplan
****************************************
Report : design
      -floorplan
Design : my_design
****************************************


-------------------------------------------------------------------------------
                        FLOORPLAN INFORMATION
-------------------------------------------------------------------------------

CORE AND CHIP AREA INFORMATION
--------------------------
```

```
Core Area is :   22267.123
Chip Area is :   22267.123
---------------------------

SITE ROW INFORMATION
-------------------------------------------------------------------------------
Site Name    Width    Height   TotRows    TotTiles        Area
unit         0.32     2.52       53       27613     22267.123
-------------------------------------------------------------------------------

BLOCKAGE INFORMATION
-------------------------------------------------
Blockage type              Count        Area
-------------------------------------------------
Hard placement                 0       0.000
Soft placement                 0       0.000
Hard macro                     0       0.000
Partial placement              0       0.000
Routing                        0       0.000
Routing for Top                0       0.000
Category                       0       0.000
RP Group                       0       0.000
Shaping                        0       0.000
Routing For DesRule            0       0.000
Placement allow BUFOnly        0       0.000
Register                       0       0.000
-------------------------------------------------

POWER DOMAIN INFORMATION
-----------------------------------------------------------------
Power Domain Name    VA Name        Primary Pwr Net Primary Gnd Net
-----------------------------------------------------------------
pd4              pd4           vdd           vss
pd_top             DEFAULT_VA     vdd           vss
-----------------------------------------------------------------

VOLTAGE AREA INFORMATION
-------------------------------------------------------------------------
VA Name      Number          Area   Util   bbox    bbox    bbox    bbox
             of shapes                     llx     lly     urx     ury
-------------------------------------------------------------------------
DEFAULT_VA          1   22267.123   1.00   0.00    0.00  166.72  133.56
pd4                 1   30671.323   0.00   0.00   75.60  145.00  133.56
-------------------------------------------------------------------------

GROUP BOUND INFORMATION
-----------------------------------------
No Group Bound exists

EXCLUSIVE MOVEBOUND INFORMATION
-------------------------------------------------------------------------------------
Bound Name       Area          Utilization    bbox_llx  bbox_lly  bbox_urx  bbox_ury
-------------------------------------------------------------------------------------
bd1              9783.480      0.568          0.00     60.00     133.00    133.56
-------------------------------------------------------------------------------------
```

```
HARD AND SOFT MOVEBOUND INFORMATION
-----------------------------------------
No Hard Or Soft MoveBound exists.

ROUTE GUIDE INFORMATION
-------------------------------------------------
Route Guide type          Count        Area
-------------------------------------------------
Extra Detour Region           0       0.000
Over icovl CellLayers         0       0.000
River Routing                 0       0.000
Area Double Via               0       0.000
Access Preference             0       0.000
Default                       0       0.000
Switched Dir Only             0       0.000
Max Patterns                  0       0.000
Others                        0       0.000
-------------------------------------------------

MULTIBIT REGISTER INFORMATION
-------------------------------------------------
No multibit cell exists

RP GROUP INFORMATION
-----------------------------------------
No RP Group exists

LAYER INFORMATION
-------------------------------------------------------------------------------
Layer Name Direction Ignored Pitch defWidth minWidth minSpacing sameNet
                                                    MinSpacing
-------------------------------------------------------------------------------
M1         Hor       NO      0.28    0.12    0.12     0.12     -0.00
M2         Ver       NO      0.32    0.14    0.14     0.14     -0.00
M3         Hor       NO      0.28    0.14    0.14     0.14     -0.00
M4         Ver       NO      0.28    0.14    0.14     0.14     -0.00
M5         Hor       NO      0.28    0.14    0.14     0.14     -0.00
M6         Ver       NO      0.28    0.14    0.14     0.14     -0.00
M7         Hor       NO      0.84    0.42    0.42     0.42     -0.00
-------------------------------------------------------------------------------
1
```

The following example reports routing information for the current block.

```
prompt> report_design -routing
****************************************
Report : design
       -routing
Design : my_design
****************************************
-------------------------------------------------------------------------------
                        ROUTING INFORMATION
-------------------------------------------------------------------------------

Total wire length = 3980818.28 micron
Total number of wires = 201345
```

Total number of contacts = 209794

FINAL WIRING STATISTICS

Metal layer	Num wires	% of total#	Wire length	% of total length
POLY1	0	0.00%	0.00	0.00%
METAL1	24737	12.29%	219157.76	5.51%
METAL2	98784	49.06%	900933.73	22.63%
METAL3	60710	30.15%	1065392.09	26.76%
METAL4	11834	5.88%	894532.03	22.47%
METAL5	4202	2.09%	657354.60	16.51%
METAL6	1078	0.54%	243448.08	6.12%

P/G Wiring Statistics

Metal layer	Num wires	% of total#	Wire length	% of total length
POLY1	0	0.00%	0.00	0.00%
METAL1	0	0.00%	0.00	0.00%
METAL2	0	0.00%	0.00	0.00%
METAL3	0	0.00%	0.00	0.00%
METAL4	147	62.03%	79167.16	34.15%
METAL5	90	37.97%	152653.63	65.85%
METAL6	0	0.00%	0.00	0.00%

Shape Pattern Wiring Statistics

Metal layer	Num shapePatterns	% of total#	Wire length	% of total length
POLY1	0	0.00%	0.00	0.00%
METAL1	0	0.00%	0.00	0.00%
METAL2	0	0.00%	0.00	0.00%
METAL3	0	0.00%	0.00	0.00%
METAL4	23	79.31%	13942.20	65.93%
METAL5	6	20.69%	7204.70	34.07%
METAL6	0	0.00%	0.00	0.00%

All the PG shape patterns stand for 369 shapes.

Horizontal/Vertical Wire Distribution

Metal layer	Hor. length	% of hor.	Ver. length	% of ver.
POLY1	0.00	0.00%	0.00	0.00%
METAL1	214713.18	11.03%	4444.58	0.22%
METAL2	18509.18	0.95%	882424.55	43.39%
METAL3	1055534.89	54.21%	9857.20	0.48%
METAL4	1100.42	0.06%	893431.61	43.93%
METAL5	657040.56	33.75%	314.04	0.02%
METAL6	52.56	0.00%	243395.52	11.97%

FINAL VIA STATISTICS

Via layer	Via def name	Count	% of layer vias
VIA12	via1(rot)	70532	78.27%
VIA12	via1	19586	21.73%

Double via conversion rate for layer VIA12 = 0.00% (0 / 90118 vias)

VIA23	via2	94149	100.00%

```
VIA23       via2(rot)                    2       0.00%
 Double via conversion rate for layer VIA23 = 0.00% (0 / 94151 vias)

VIA34       via3                     15939      99.95%
VIA34       via3(rot)                    8       0.05%
 Double via conversion rate for layer VIA34 = 0.00% (0 / 15947 vias)

VIA45       via4                      6389      97.41%
VIA45       via4(rot)                   11       0.17%
VIA45       via4_67X67                   8       0.12%
VIA45       via4_67X21                  60       0.91%
VIA45       via4_21X67                  91       1.39%
 Double via conversion rate for layer VIA45 = 2.42% (159 / 6559 vias)

VIA56       via5                      1996     100.00%
 Double via conversion rate for layer VIA56 = 0.00% (0 / 1996 vias)

Overall double via conversion rate = 0.08%

Custom Via Statistics

Via def name                  Count
via4_9740_9640_ALL_19_19          16
via3_10780_19520_ALL_21_38         8
via3_10780_9640_ALL_21_19         45
via3_10780_7040_ALL_21_14          6
via3_380_9640_ALL_1_19             2
via4_10780_9640_ALL_21_19         48
via4_9740_10680_ALL_19_21         96
via4_19620_10680_ALL_38_21       165
via4_10780_10680_ALL_21_21       360
via3_19620_10680_ALL_38_21       165
via3_10260_10680_ALL_20_21         4
via4_380_10680_ALL_1_21            2
via3_4540_10680_ALL_9_21           2
via4_380_4960_ALL_1_10             1
via4_19620_4960_ALL_38_10          1
via3_4540_4960_ALL_9_10            1
via3_19620_4960_ALL_38_10          1
via4_6620_10680_ALL_13_21         10
via3_6620_10680_ALL_13_21         10
via3_9740_10680_ALL_19_21         32
via4_10260_10680_ALL_20_21         1
via3_9220_10680_ALL_18_21          1
via3_4540_9640_ALL_9_19            7
via4_19100_10680_ALL_37_21         4
via3_19100_10680_ALL_37_21         4
via4_12860_10680_ALL_25_21         2
via3_12860_10680_ALL_25_21         2
via4_8700_10680_ALL_17_21         10
via3_8700_10680_ALL_17_21         10
via4_2980_10680_ALL_6_21           6
via3_8180_10680_ALL_16_21          1
1
```

SEE ALSO

report_clocks(2)
report_ports(2)

report_design_mismatch

Reports mismatch summary for the specified designs and reference libraries. By default reports information from current design and its reference library. More detailed information of 'any specific design[s]', 'any specific library[ies]' or 'all designs from current library and its reference libraries' can be reported through various available options. By default reports all mismatches except accepted.

SYNTAX

status **report_design_mismatch**
[-mismatch_type *type*]
[-verbose]
[-repair_status repaired | not_repaired | accepted | deleted | ignored]
[-nosplit]
[objects]

Data Types

type list

ARGUMENTS

-mismatch_type

Specifies the list of mismatch types to report mismatches. Specify one or more of the following values: **bus_bit_naming**, **lib_missing_logical_port**, **lib_missing_physical_port**, **lib_missing_physical_reference**, **lib_port_name_case**, **lib_cell_name_case**, **lib_port_direction**, **lib_port_name_synonym**, **lib_port_type**, **lib_missing_rail_pg**, **lib_port_missing_via_region**, **lib_port_missing_access_edge**, **missing_logical_reference**, **missing_port**, **port_name_case**, **reference_name_case**, **port_name_synonym**, **macro_cell_contain_too_many_obs**, **exceed_ndr_limitation**, **layer_missing_prefer_direction**, and **empty_logic_module**.

-verbose

Displays detailed information about each mismatch type.

-repair_status

Specifies the list of mismatch states to report mismatches.

-nosplit

Does not split lines if column overflows.

objects

List of designs/libs/lib_cells/mismatch_objects to report mismatches.These objects can be output from get_designs, get_libs, get_lib_cells and get_mismatch_objects command .

DESCRIPTION

This command reports mismatch summary for the current design and its reference libraries. The summarized report will have count of mismatch objects under each mismatch type. The report contains all mismatches except those having repair state as accepted. Accepted mismatches are considered differently than the other mismatches, and thus are not reported by default. They can be reported using **repair_status** option. Specific libraries/designs/library cells/mismatch objects can also be specified explicitly to report mismatches upon. List of libraries/designs/library cells/mismatch objects can be supplied via get_libs/get_designs/get_lib_cells/get_mismatch_objects commands respectively. When **verbose** option is used, this will report detailed information for each mismatch object under each mismatch type for the current design. When **repair_status** option is used, this command will only report mismatches with specified mismatch repair status. Valid values for mismatch repair status are: repaired, accepted, ignored, deleted and not_repaired. When **mismatch_type** option is used, this command will report mismatches for specified mismatch type.

EXAMPLES

The following example reports mismatches:

```
prompt> report_design_mismatch
****************************************
Report : Reporting Mismatches
Version: J-2014.12-SP1
Date   : Thu Jan 15 22:59:43 2015
****************************************


==============================
Design : ORCA
==============================
Mismatch Type                 Total Count   Repaired  Unrepaired
--------------------------------------------------------------------------------
port_name_case                 2          2       0
1

==============================
Library : SPREG_SVA
==============================
Mismatch Type                 Total Count   Repaired  Unrepaired
--------------------------------------------------------------------------------
lib_missing_physical_reference      3          3       0

1
```

The following example reports each mismatch in detail:

```
prompt> report_design_mismatch -verbose
****************************************
Report : Reporting mismatches
Version: J-2014.12-SP1
Date   : Wed Dec  3 02:59:59 2014
****************************************
```

```
==============================
Design : ORCA
==============================
Mismatch Type                  Total Count   Repaired  Unrepaired
--------------------------------------------------------------------------------
port_name_case                  2           2       0


Mismatch Type : port_name_case
ObjName      ObjType  State        Repair Strategy                 Inst
  MMObjName
--------------------------------------------------------------------------------
bufbdf/I     port    repaired     record_port_name_case_insensitivity  6
  port_name_case_ORCA_bufbdf/I
invbd2/I     port    repaired     record_port_name_case_insensitivity  24
  port_name_case_ORCA_invbd2/I
1
```

The following example reports mismatches for a specific design:

```
prompt> report_design_mismatch [get_designs REG_FILE]
****************************************
Report : Reporting mismatches
Version: J-2014.12-SP1
Date   : Wed Dec  3 02:59:59 2014
****************************************
==========================
Design: REG_FILE
==========================

Mismatch Type                  Total Count   Repaired  Unrepaired
--------------------------------------------------------------------------------
port_name_case                  1           1       0

1
```

The following example reports verbose information of mismatches on a specific design:

```
prompt> report_design_mismatch [get_designs REG_FILE] -verbose
****************************************
Report : Reporting mismatches
Version: J-2014.12-SP1
Date   : Wed Dec  3 02:59:59 2014
****************************************
==============================
Design : REG_FILE
==============================
Mismatch Type                  Total Count   Repaired  Unrepaired
--------------------------------------------------------------------------------
port_name_case                  1           1       0


Mismatch Type : port_name_case
ObjName      ObjType  State        Repair Strategy                 Inst
  MMObjName
--------------------------------------------------------------------------------
inv0d1/I     port    repaired     record_port_name_case_insensitivity  1
```

```
  port_name_case_REG_FILE_inv0d1/I
1
```

The following example reports mismatches on all designs:

```
prompt> report_design_mismatch [get_designs]
****************************************
Report : Reporting mismatches
Version: J-2014.12-SP1
Date   : Wed Dec  3 02:59:59 2014
****************************************

==========================
Design: ORCA
==========================

Mismatch Type                 Total Count   Repaired  Unrepaired
--------------------------------------------------------------------------------
port_name_case                  1           1         0

==========================
Design: REG_FILE
==========================

Mismatch Type                 Total Count   Repaired  Unrepaired
--------------------------------------------------------------------------------
port_name_case                  1           1         0
1
```

The following example reports mismatches for all libraries:

```
prompt> report_design_mismatch [get_libs]
****************************************
Report : Reporting mismatches
Version: J-2014.12-SP1
Date   : Wed Dec  3 02:59:59 2014
****************************************

==================
Library: bb
==================
Mismatch Type                 Total Count   Repaired  Unrepaired
--------------------------------------------------------------------------------
port_name_case                  3           3         0

1
```

The following example reports verbose information of mismatches on all libraries:

```
prompt> report_design_mismatch [get_libs] -verbose
****************************************
Report : Reporting mismatches
Version: J-2014.12-SP1
Date   : Wed Dec  3 02:59:59 2014
****************************************

==============================
Library : bb
==============================
Mismatch Type                 Total Count   Repaired  Unrepaired
--------------------------------------------------------------------------------
```

```
port_name_case                  3          3       0


Mismatch Type : port_name_case
ObjName      ObjType  State       Repair Strategy                  Inst
  Design           MMObjName
-------------------------------------------------------------------------------
inv0d1/I     port    repaired     record_port_name_case_insensitivity  1
  REG_FILE          port_name_case_REG_FILE_inv0d1/I
bufbdf/I     port    repaired     record_port_name_case_insensitivity  6
  ORCA             port_name_case_ORCA_bufbdf/I
invbd2/I     port    repaired     record_port_name_case_insensitivity  24
  ORCA             port_name_case_ORCA_invbd2/I
1
```

The following example reports mismatches with repaired status:

```
prompt> report_design_mismatch -repair_status repaired
****************************************
Report : Reporting mismatches
Version: J-2014.12-SP1
Date   : Wed Dec  3 02:59:59 2014
****************************************
============================
Design: ORCA
============================

Mismatch Type                   Repaired
-------------------------------------------------------------------------------
port_name_case                   2
1
```

The following example reports mismatches with accepted status:

```
prompt> report_design_mismatch -repair_status accepted
****************************************
Report : Reporting mismatches
Version:
Date   :
****************************************
============================
Design: ORCA
============================

Mismatch Type                   Accepted
-------------------------------------------------------------------------------
port_name_case                   2
1
```

The following example accepts a list of mismatch states:

```
prompt> report_design_mismatch -repair_status {repaired not_repaired}
****************************************
Report : Data Mismatches
Version:
Date   :
****************************************
```

```
==============================
Design : top
==============================
Mismatch Type                  Unrepaired  Repaired
--------------------------------------------------------------------------------
port_name_case                  1          0
1
```

The following example reports mismatches for specified mismatch types:

```
prompt> report_design_mismatch -mismatch_type [get_mismatch_types port_name_case] -verbose
****************************************
Report : Reporting mismatches
Version: J-2014.12-SP1
Date   : Wed Dec  3 02:59:59 2014
****************************************
==============================
Design : ORCA
==============================
Mismatch Type                  Total Count   Repaired  Unrepaired
--------------------------------------------------------------------------------
port_name_case                  2           2        0


Mismatch Type : port_name_case
ObjName      ObjType  State        Repair Strategy                  Inst
  MMObjName
--------------------------------------------------------------------------------
bufbdf/I     port    repaired     record_port_name_case_insensitivity  6
  port_name_case_ORCA_bufbdf/I
invbd2/I     port    repaired     record_port_name_case_insensitivity  24
  port_name_case_ORCA_invbd2/I
1
```

SEE ALSO

get_mismatch_types(2)
get_mismatch_objects(2)
create_mismatch_config(2)
report_mismatch_configs(2)
set_current_mismatch_config(2)
get_current_mismatch_config(2)

report_design_rules

Displays design_rule's information from the current library's technology.

SYNTAX

string **report_design_rules**
[-tech *tech_object*]
[-library *library*]
[-verbose]
[-nosplit]
[-all]
[*pattern*]

Data Types

tech	collection
library	string or collection
patterns	string or collection

ARGUMENTS

-tech *tech_object*

Specifies the technology object from which to report the design rules.

The **-tech** and **-library** options are mutually exclusive; you can specify only one. If you do not specify either option, the rules are reported from the technology object of the current library.

-library *library*

Specifies the library from which to report the design rules. The command reports the rules from the technology object associated with the specified library. In an implementation tool, the library is a design library. In the library manager, the library is a cell library. You can specify the library using the library's name or a collection that contains the library.

The **-tech** and **-library** options are mutually exclusive; you can specify only one. If you do not specify either option, the rules are reported from the technology object of the current library.

-verbose

Display all the properties of design rule. Without **-verbose**, only layer1, layer2 and min_spacing properties will be displayed.

-nosplit

Prevents line splitting if the column overflows. Most information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting

and facilitates writing software to extract information from the report output.

-all

Report all design rules.

patterns

Displays information only on the specified design_rule objects. *patterns* contains either a collection of design_rules or a pattern that matches the design_rule names.

DESCRIPTION

This command displays information about design_rules in the current libray's technology.

EXAMPLES

Below example outputs the report for all design_rule objects.

```
prompt> report_design_rules -all
****************************************
Report : report_design_rules
Version: M-2016.12-SP4-BETA
Date   : Mon Apr  3 05:48:28 2017
****************************************
Name            Property Name          Value
------------------------------------------------------------------------------
DESIGN_RULE_0     layer1               via1Blockage
                  layer2               VIA1
                  min_spacing          0

DESIGN_RULE_1     layer1               via2Blockage
                  layer2               VIA2
                  min_spacing          0

1
```

Below example outputs the design_rule named DESIGN_RULE_0

```
prompt> report_design_rules DESIGN_RULE_0
****************************************
Report : report_design_rules
Version: M-2016.12-SP4-BETA
Date   : Mon Apr  3 05:50:57 2017
****************************************
Name            Property Name          Value
------------------------------------------------------------------------------
DESIGN_RULE_0     layer1               via1Blockage
                  layer2               VIA1
                  min_spacing          0
```

1

SEE ALSO

get_design_rules(2)
remove_pr_rules(2)
create_pr_rule(2)
report_hierarchy(2)
report_nets(2)
report_cells(2)
report_references(2)

report_device_group

Reports statistics on cell count and area by device group names.

SYNTAX

int **report_device_group**
[-verbose]
[-significant_digits *digits*]
[-nosplit]
[-detailed *cell_type*]
[-datapath_cells_only]
[-standard_cells_only]
[-hierarchy]
[-hier_device_groups *group_list*]
[-hier_threshold *digits*]
[*cell_list*]

Data Types

digits int
cell_type list
cell_list list
group_list list

ARGUMENTS

-verbose

Lists all of the cells belonging to each device group. If this option is omitted, the default report shows only the number and percentage of cells.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 through 13. The default is 2. Using this option overrides the value set by the **report_default_significant_digits** variable.

-nosplit

Prevents line splitting and facilitates writing scripts to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-detailed *cell_type*

Specifies the cell types for which detailed report is to be generated. Acceptable types are repeater, register and clock_network.

-datapath_cells_only

Reports only for cells in the data path excluding clock_network, macros and blackboxes

-standard_cells_only

Reports only for cells in data path and clock network

-hierarchy

Prints the hierarchical report for all device groups present in the design

-hier_device_groups *group_list*

Prints the hierarchical report for user specified device groups in the design.

hier_threshold *digits*

Limits the report to hierarchical cells that has leaf cells more than the specified threshold

cell_list

Specifies a list of cells on which to report the device groups. If not specified, all cells in the **current_design** are considered.

DESCRIPTION

This command reports the number, area, and percentage of cells in each device group for the specified list of cells or designs. The report also shows separately, the number, area, and percentage of repeater, combinational, register, sequential, clock_network, macro and physical_only cells in each group.

The device group is defined in the technology libraries with the **device_group** attribute on the cell. You can also set the attributes on the objects using the **set_attribute** command. The attributes can be any string values, but the values must be related to the device group of the cell to be meaningful. Any cells in the design that have no device group attribute are included in the "undefined" group. Cells in the abstract block will not be reported.

EXAMPLES

This command generates a summary of cells in each device group:

```
prompt> report_device_group
****************************************
Report : device_group
Design : ChipTop
Version: Q-2019.12-SP2-DEV
Date   : Wed Feb 19 04:02:44 2020
****************************************

Cell Count Report
---------------------------------------------------------------------------------------------------------------------------------------------------------
Group Names            All (%)      Repeaters (%)  Combinational (%)     Registers (%)     Sequentials (%)  Clock_network (%)
---------------------------------------------------------------------------------------------------------------------------------------------------------
```

```
Narrow1          1  (16.67%)      0   (0.00%)      1  (16.67%)      0   (0.00%)      0   (0.00%)      0   (0.00%)      0   (0.00%)
Wide1            5  (83.33%)      3  (50.00%)      0   (0.00%)      1  (16.67%)      0   (0.00%)      1  (16.67%)      0   (0.00%)
--------------------------------------------------------------------------------------------------------------------------------------
Total            6 (100.00%)      3  (50.00%)      1  (16.67%)      1  (16.67%)      0   (0.00%)      1  (16.67%)      0   (0.00%)
--------------------------------------------------------------------------------------------------------------------------------------

Cell Area Report
--------------------------------------------------------------------------------------------------------------------------------------

Group Names           All (%)      Repeaters (%)  Combinational (%)     Registers (%)    Sequentials (%)  Clock_network (%)
--------------------------------------------------------------------------------------------------------------------------------------
Narrow1          5.60  (10.00%)    0.00   (0.00%)    5.60  (10.00%)    0.00   (0.00%)    0.00   (0.00%)    0.00   (0.00%)    0.00  (0.
Wide1           50.40  (90.00%)   10.08  (18.00%)    0.00   (0.00%)   28.00  (50.00%)    0.00   (0.00%)   12.32  (22.00%)    0.00
--------------------------------------------------------------------------------------------------------------------------------------
Total           56.00 (100.00%)   10.08  (18.00%)    5.60  (10.00%)   28.00  (50.00%)    0.00   (0.00%)   12.32  (22.00%)    0.00
--------------------------------------------------------------------------------------------------------------------------------------

1

prompt> report_device_group -detailed repeater

****************************************
Report : device_group
Design : ChipTop
Version: Q-2019.12-SP2-DEV
Date   : Wed Feb 19 04:11:41 2020
****************************************

Cell Count Report
--------------------------------------------------------------------------------------------------------------------------------------

Group Names           All (%)      Repeaters (%)  Combinational (%)     Registers (%)    Sequentials (%)  Clock_network (%)
--------------------------------------------------------------------------------------------------------------------------------------
Narrow1          1  (16.67%)      0   (0.00%)      1  (16.67%)      0   (0.00%)      0   (0.00%)      0   (0.00%)      0   (0.00%)
Wide1            5  (83.33%)      3  (50.00%)      0   (0.00%)      1  (16.67%)      0   (0.00%)      1  (16.67%)      0   (0.00%)
--------------------------------------------------------------------------------------------------------------------------------------
Total            6 (100.00%)      3  (50.00%)      1  (16.67%)      1  (16.67%)      0   (0.00%)      1  (16.67%)      0   (0.00%)
--------------------------------------------------------------------------------------------------------------------------------------

Repeater detailed Cell Count Report
------------------------------------------------------------------------------------------

Group Names           All (%)        Buffer (%)       Inverter (%)     Hold buffer (%)
------------------------------------------------------------------------------------------
Narrow1          0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0.00%)
Wide1            3 (100.00%)       0   (0.00%)       3 (100.00%)       0   (0.00%)
------------------------------------------------------------------------------------------
Total            3 (100.00%)       0   (0.00%)       3 (100.00%)       0   (0.00%)
------------------------------------------------------------------------------------------

Cell Area Report
--------------------------------------------------------------------------------------------------------------------------------------

Group Names           All (%)      Repeaters (%)  Combinational (%)     Registers (%)    Sequentials (%)  Clock_network (%)
```

```
---------------------------------------------------------------------------------------------------------------------------------------------------------
Narrow1          5.60  (10.00%)    0.00   (0.00%)    5.60  (10.00%)    0.00   (0.00%)    0.00   (0.00%)    0.00   (0.00%)    0.00  (0.
Wide1           50.40  (90.00%)   10.08  (18.00%)    0.00   (0.00%)   28.00  (50.00%)    0.00   (0.00%)   12.32  (22.00%)    0.00
---------------------------------------------------------------------------------------------------------------------------------------------------------
Total           56.00 (100.00%)   10.08  (18.00%)    5.60  (10.00%)   28.00  (50.00%)    0.00   (0.00%)   12.32  (22.00%)    0.00
---------------------------------------------------------------------------------------------------------------------------------------------------------

Repeater detailed Cell Area Report
-------------------------------------------------------------------------------------------

Group Names             All (%)        Buffer (%)       Inverter (%)     Hold buffer (%)
-------------------------------------------------------------------------------------------
Narrow1          0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)
Wide1           10.08 (100.00%)     0.00   (0.00%)    10.08 (100.00%)     0.00   (0.00%)
-------------------------------------------------------------------------------------------
Total           10.08 (100.00%)     0.00   (0.00%)    10.08 (100.00%)     0.00   (0.00%)
-------------------------------------------------------------------------------------------

1

prompt> report_device_group -hierarchy -hier_threshold 1

****************************************
Report : device_group
Design : ChipTop
Version: Q-2019.12-SP2-DEV
Date   : Wed Feb 19 04:13:37 2020
****************************************
Hierarchy               |         Total Cell Detail         |                Narrow1                |                 Wide1                 |
                        |     Cell#          Area Ratio% |      Cell#          Area Block% Design% |      Cell#          Area Block% Desig
---------------------------------------------------------------------------------------------------------------------------------------------------------
b1                      |        6        5.60e+01 100.00 |         1        5.60e+00  10.00   10.00 |         5        5.04e+01  90.00   90.00
  u1                    |        1        3.36e+00   6.00 |         0        0.00e+00   0.00    0.00 |         1        3.36e+00 100.00    6.00 |
  u2                    |        1        3.36e+00   6.00 |         0        0.00e+00   0.00    0.00 |         1        3.36e+00 100.00    6.00 |
  u3                    |        3        3.70e+01  66.00 |         1        5.60e+00  15.15   10.00 |         2        3.14e+01  84.85   56.00
    u4                  |        1        3.36e+00   6.00 |         0        0.00e+00   0.00    0.00 |         1        3.36e+00 100.00    6.00 |

1
```

SEE ALSO

set_attribute(2)

report_dff_connections

Reports connections found using **compute_dff_connections**.

SYNTAX

status **report_dff_connections**
[-max_reg *register_count*]
[-max_gate *gate_count*]
[-min_connections *connection_count*]
[-inst_to_inst_min *connection_count*]
[-port_to_port_min *connection_count*]
[-inst_to_port_min *connection_count*]
[-port_to_inst_min *connection_count*]

[-from_blocks *list_of_designs*]
[-to_blocks *list_of_designs*]
[-from_macros *list_of_macros*]
[-to_macros *list_of_macros*]
[-from_bounds *list_of_bounds*]
[-to_bounds *list_of_bounds*]
[-from_ports *list_of_ports*]
[-to_ports *list_of_ports*]

[-format text | html | csv] (default: text)
[-filename *output_file*]
[-overwrite]

[-no_blocks]
[-all_macros]
[-all_bounds]
[-all_ports]
[-same_types_only]
[-no_legend]
[-reduce_width]
[-include_empty]

Data Types

register_count int
gate_count int
connection_count int
list_of_designs collection or list of names of designs (blocks or macros)
list_of_bounds collection or list of names of module boundaries
list_of_ports collection or list of names of ports
output_file string

ARGUMENTS

-max_reg *register_count*

Specifies the maximum number of registers a path can contain and still be reported. It is an error to specify a larger value than used during **compute_dff_connections**.

-max_gate *gate_count*

Specifies the maximum number of gates a path can contain and still be reported. It is an error to specify a larger value than used during **compute_dff_connections**.

-min_connections *connection_count*

Only include blocks/macros/bounds/ports in the report with this many or more connections to another block/macro/bound or port. Specifying a large value will report less total connections. By default, this value is 0 (i.e. report all connection counts).

-inst_to_inst_min *connection_count*

A subset of **-min_connections** - only applies to the reporting of inst to inst connections (blocks/bounds/macros to blocks/bounds/macros).

-inst_to_port_min *connection_count*

A subset of **-min_connections** - only applies to the reporting of inst to port connections (blocks/bounds/macros to ports).

-port_to_inst_min *connection_count*

A subset of **-min_connections** - only applies to the reporting of port to inst connections (ports to blocks/bounds/macros).

-port_to_port_min *connection_count*

A subset of **-min_connections** - only applies to the reporting of port to port connections (ports to ports).

-no_blocks

Do not report on any blocks. **Note: Default is to report on all blocks, as if -all_blocks was specified, although this is not an actual parameter.**

-from_blocks *list_of_designs*

Only report block connections starting at these blocks. Cannot be specified together with **-no_blocks** and **-to_blocks**. If not specified, connections from all blocks will be reported.

-to_blocks *list_of_designs*

Only report block connections ending at these blocks. Cannot be specified together with **-no_blocks** and **-from_blocks**. If not specified, connections to all blocks will be reported.

-all_bounds

Report to/from all bounds (module boundaries). Default is no bounds included in report.

-from_bounds *list_of_designs*

Only report bound (module boundary) connections starting at these bounds. Cannot be specified with **-all_bounds** and **-**

to_bounds.

-to_bounds ***list_of_designs***

Only report bound (module boundary) connections ending at these bounds. Cannot be specified with **-all_bounds** and **-from_bounds**.

-all_macros

Report to/from all macros. Default is no macros included in report.

-from_macros ***list_of_macros***

Only report macro connections starting at these macros. Cannot be specified with **-all_macros** and **-to_macros**.

-to_macros ***list_of_macros***

Only report macro connections ending at these macros. Cannot be specified with **-all_macros** and **-from_macros**.

-all_ports

Report to/from all ports. Default is no ports included in report.

-from_ports ***list_of_ports***

Only report port connections starting at these ports. Cannot be specified with **-all_ports** and **-to_ports**.

-to_ports ***list_of_ports***

Only report port connections ending at these ports. Cannot be specified with **-all_ports** and **-from_ports**.

-format text | **html** | **csv**

Produce report in this output format. Default is text (ASCII). HTML = Hypertext Markup Language. CSV = Comma Separated Values.

-filename ***output_file***

Write report to this file. Default is to the console/terminal.

-overwrite

If specified, overwrite an existing report file.

-same_types_only

Only report between identical types. I.e. block-to-block, macro-to-macro, bound-to-bound and port-to-port. Which sub-reports are actually produced is influenced by other settings.

-no_legend

Instead of a legend, use full names in the report row/column headings. Will result in much longer rows (wider columns), but may be preferable for parsing the output.

-reduce_width

Try to produce as little whitespace as possible in report columns. This will result in columns of differing widths (especially if used with **-no_legend**). Useful for viewing of the report, but may make parsing the report more difficult.

-include_empty

Include instances and ports with no to/from connections in the report. This will report the connections between all blocks, for example, not only those that actually have connections.

DESCRIPTION

This command reports the results from the Data Flow Flylines (DFF) tracer (**compute_dff_connections**). A GUI visualization tool is available for viewing DFF results. This command allows for reporting the data in a structured manner. This command can report connections between blocks, macros, bounds and ports. It can produce output in text format (ASCII), HTML or CSV and display it to the console or save it to a file. Various options are available to limit output to the desired subset of DFF data.

By default, connections to/from all blocks are shown. This can be altered using **-no_blocks**, **-from_blocks** and **-to_blocks**, although all three parameters may not be specified together. If only **-from_blocks** is specified, then the "to blocks" list defaults to all blocks. If **-from_blocks** and **-no_blocks** is specified, then only connections from the given list of blocks is reported, and no connections to blocks are reported.

The other parameter combinations work similarly. **-all_macros** by itself reports to/from all macros. **-all_macros** and **-from_macros** reports from the given macros and to all macros. **-all_macros** and **-to_macros** reports from all macros and to the given macros. All three macro parameters may not be specified together.

NOTES

Specifying **-min_connections** (or the other connection count restrictions) will often show connections with a lower value than the specified value. For example, if block A has 10 connections to block B and 20 connections to block C, **-min_connections 15** is required to include block A. If block D has 25 connections to block B, block B may be included in the overall "TO" column, even though it only has 10 connections from block A. The choice is either to show - (no connections) from A to B, which would be incorrect, or "X" (not shown), or 10. For the sake of accuracy, 10 is shown.

EXAMPLES

The following example traces the current design using default constraints, and then reports the results of all block-to-block connections to the screen using default settings.

```
prompt> compute_dff_connections
prompt> report_dff_connections
```

SEE ALSO

compute_dff_connections(2)

report_dft

Reports DFT configuration for the current session.

SYNTAX

string **report_dft**
[-scan_configuration]
[-dft_signal]
[-dft_drc_configuration]
[-scan_compression_configuration]
[-logicbist_configuration]
[-partitions]
[-wrapper_specification]
[-clock_controller]
[-testability]
[-scan_path]
[-pipeline]
[-serialize]
[-test_assume]
[-clock_gating_pin]
[-scan_group]
[-location]
[-dft_configuration]
[-ieee_1500_configuration]
[-dft_clock_gating_configuration]
[-nosplit]

ARGUMENTS

-scan_configuration

Displays the current scan configuration settings. Use the **set_scan_configuration** command to modify them.

-dft_signal

Displays port, signal type, active state, hookup pin usage, and timing information for all DFT signals. Use the **set_dft_signal** command to modify them.

-dft_drc_configuration

Displays the current DFT DRC settings. Use the **set_dft_drc_configuration** command to modify them.

-scan_compression_configuration

Displays the current scan compression configuration settings. Use the **set_scan_compression_configuration** command to

modify them.

-logicbist_configuration

Displays the current LogicBIST self-test configuration settings. Use the **set_logicbist_configuration** command to modify them.

-partitions

Displays the current DFT partition settings. Use the **define_dft_partition** command to specify them.

-wrapper_specification

Displays the current core wrapping settings. Use the **set_wrapper_configuration** command to modify them.

-clock_controller

Displays the current on-chip clocking (OCC) controller settings. Use the **set_dft_clock_controller_configuration** command to modify them.

-testability

Displays the current automatically inserted test point settings. Use the **set_testability_configuration** command to modify them.

-scan_path

Displays the current user-defined scan path settings. Use the **set_scan_path** command to modify them.

-pipeline

Displays the current pipelined scan data settings. Use the **set_pipeline_scan_data_configuration** command to modify them.

-serialize

Displays the current serializer settings. Use the **set_serialize_configuration** command to modify them.

-test_assume

Displays the current DRC assumed-constant settings. Use the **set_test_assume** command to modify them.

-clock_gating_pin

Displays the current user-identified clock-gating test pin settings. Use the **set_dft_clock_gating_pin** command to modify them.

-scan_group

Displays the current scan group settings. Use the **set_scan_group** command to modify them.

-location

Displays the current DFT insertion location settings. Use the **set_dft_location** command to modify them.

-dft_configuration

Displays the current DFT client-enable settings. Use the **set_dft_configuration** command to modify them.

-ieee_1500_configuration

Displays the current IEEE 1500 settings. Use the **set_ieee_1500_configuration** command to modify them.

-dft_clock_gating_configuration

Displays the current DFT clock-gating settings. Use the **set_dft_clock_gating_configuration** command to specify them.

-nosplit

Does not split lines if column overflows.

DESCRIPTION

This command displays information about the DFT features that have been set in the current session. You can generate specific reports by using the command's options. By default, all reports are displayed.

EXAMPLES

The following example displays a report for the scan configuration in the current session.

```
prompt> report_dft -scan_configuration
****************************************
Report : Scan configuration
 ...

****************************************


----------------------------------------
TEST MODE: COMPRESS
VIEW    : Specification
----------------------------------------
Chain count:                    Unspecified
Maximum scan chain length:          10
Insert terminal lockup:             False
```

The following example gives a report for the scan compression configuration settings in the current design.

```
prompt> report_dft -scan_compression_configuration
****************************************
Report : Scan Compression Configuration
 ...

****************************************


----------------------------------------
TEST MODE: COMPRESS
VIEW    : Specification
----------------------------------------
xtolerance                      0
Inputs                          8
Outputs                         8
Chain Count                     9
Maximum Chain Length                10
```

The following example gives a report for the DFT signals in the current design.

```
prompt> report_dft -dft_signal
****************************************
```

```
Report : DFT signals
 ...

****************************************

-----------------------------------------
TEST MODE: all_dft
VIEW    : Specification
-----------------------------------------
Port          SignalType       Active   Hookup   Usage        Timing
----------    ----------       ------   ------   -----        ------
SE            ScanEnable       -        -        Scan
SI            ScanDataIn       -        U16/Z    Scan
tsi0          ScanDataIn       -        -        ClockGating
SO            ScanDataOut      -        U16/B    Scan
tso0          ScanDataOut      -        -        Scan

-----------------------------------------
TEST MODE: all_dft
VIEW    : Existing DFT
-----------------------------------------
Port          SignalType       Active   Hookup   Usage        Timing
----------    ----------       ------   ------   -----        ------
clk           ScanMasterClock  1        -        -            P 100.0 R 45.0 F 55.0
sig1          Constant         0        -        -
rst           Reset            0        -        -            P 100.0 R 55.0 F 45.0
U12/B         Reset            1        -        -            P 100.0 R 45.0 F 55.0
```

SEE ALSO

set_scan_compression_configuration(2)
set_dft_signal(2)
set_scan_configuration(2)

report_dft_clock_controller

Reports the on-chip clocking controller specification for the current design.

SYNTAX

status **report_dft_clock_controller**

ARGUMENTS

The **report_dft_clock_controller** command has no arguments.

DESCRIPTION

This command displays the current on-chip clocking controller specification applied to the current design.

Use the **set_dft_clock_controller** command to specify an on-chip clocking controller specification.

For more information, see the man page for the **set_dft_clock_controller** command.

EXAMPLES

The following command reports the clock_controller controller specification for the current design.

```
prompt> report_dft_clock_controller

****************************************
Report : Clock controller
Design : top
Version: G-2012.06-SP3
Date   : Tue Nov 13 12:46:05 2012
****************************************

========================================
TEST MODE: all_dft
VIEW     : Specification
========================================
```

```
Cell name:              CORE
Design:                 snps_clk_mux
Chain count:            1
Cycle count:            2
PLL clock:              UPLL/CLKO
ATE clock:              ATECLK

1
```

SEE ALSO

set_dft_clock_controller(2)

report_dft_clock_gating_configuration

Displays the options specified by the **set_dft_clock_gating_configuration** command.

SYNTAX

status **report_dft_clock_gating_configuration**

ARGUMENTS

This command takes no arguments.

DESCRIPTION

This command displays the options that were specified with the **set_dft_clock_gating_configuration** command on the current design.

EXAMPLES

The following example shows the report generated by the **report_dft_clock_gating_configuration** command:

```
prompt> set_dft_clock_gating_configuration -exclude_elements \
   {U_block_a/cg1 U_block_b/cg1}
Accepted clock gating configuration specification
1
prompt> report_dft_clock_gating_configuration
****************************************
Report : Clock Gating Configuration
Design : top
Version: P-2019.03
Date   : Thu Feb  7 06:38:49 2019
****************************************

 Clock Gating Exclude elements:       U_block_a/cg1/
                                      U_block_b/cg1/
```

SEE ALSO

set_dft_clock_gating_configuration(2)

report_dft_clock_gating_pin

Displays the specification specified by the **set_dft_clock_gating_pin** command.

SYNTAX

status **report_dft_clock_gating_pin**

ARGUMENTS

This command takes no arguments.

DESCRIPTION

This command displays the options that were specified with the **set_dft_clock_gating_pin** command on the current design.

When the reported test pin of a clock-gating cell is active low and controlled by a ScanEnable or TestMode signal, the control signal is reported as ScanEnable_inverted or TestMode_inverted, respectively.

EXAMPLES

The following example shows the report generated by the **report_dft_clock_gating_pin** command. Consider a set of **set_dft_clock_gating_pin** commands as follows:

```
set_dft_clock_gating_pin {sub1/clk_gate_out1_reg} -pin_name TE \
  -control_signal ScanEnable -active_state 1

set_dft_clock_gating_pin {sub2/clk_gate_out1_reg} -pin_name TE \
  -control_signal ScanEnable -active_state 0

set_dft_clock_gating_pin {sub3/clk_gate_out1_reg} -pin_name TE \
  -control_signal TestMode -active_state 1

set_dft_clock_gating_pin {sub4/clk_gate_out1_reg} -pin_name TE \
  -control_signal TestMode -active_state 0
```

The output of the **report_dft_clock_gating_pin** command for these commands is:

```
prompt> report_dft_clock_gating_pin
****************************************
Report : DFT Clock gating pin configuration
Design : top
Version: P-2019.03
Date   : Thu Feb  7 06:45:35 2019
****************************************

User Identified Clock Gating Cells: 4
------------------------------------------------------------------------
Control Signal      : Pin           : Cell
------------------------------------------------------------------------
ScanEnable          : TE            : sub1/clk_gate_out1_reg
ScanEnable_inverted : TE            : sub2/clk_gate_out1_reg
TestMode            : TE            : sub3/clk_gate_out1_reg
TestMode_inverted   : TE            : sub4/clk_gate_out1_reg
```

SEE ALSO

set_dft_clock_gating_pin(2)

report_dft_configuration

Displays the options specified by the **set_dft_configuration** command.

SYNTAX

status **report_dft_configuration**

ARGUMENTS

The **report_dft_configuration** command has no arguments.

DESCRIPTION

This command displays options specified by the **set_dft_configuration** command on the current design.

The command shows which DFT clients are enabled or disabled.

EXAMPLES

The following is an example of the report generated by the **report_dft_configuration** command:

```
prompt> report_dft_configuration

****************************************
Report : DFT Configuration
Design : top
Version: P-2019.03
Date   : Thu Feb  7 06:47:59 2019
****************************************

DFT Structures                Status
--------------               --------
Scan:                        enable
Testability:                  disable
Wrapper:                      enable
Scan Compression:                   enable
```

```
    Pipeline Scan Data:          enable
    Mode Decoding Style:         binary
```

SEE ALSO

set_dft_configuration(2)

report_dft_drc_configuration

Displays the options specified by the **set_dft_drc_configuration** command.

SYNTAX

status **report_dft_drc_configuration**

ARGUMENTS

The **report_dft_drc_configuration** command has no arguments.

DESCRIPTION

This command displays options specified by the **set_dft_drc_configuration** command on the current design.

EXAMPLES

The following is an example of the report generated by the **report_dft_drc_configuration** command:

```
prompt> report_dft_drc_configuration

****************************************
Report : dft_drc configuration
Design : top
Version: P-2019.03
Date   : Mon Mar  4 11:59:15 2019
****************************************

----------------------------------------
TEST MODE: Internal
VIEW     : Specification
----------------------------------------
Run DRC                          True
Generate Under-the-Hood Library  True
Internal Pins                    Enabled
GTECH Cells Used                 False
```

```
Override User-Specified Library       False
Save Under-the-Hood Library           False
Use Test Model                        True
Pll Bypass                            False
Preform Static- X Analysis            False
```

SEE ALSO

set_dft_drc_configuration(2)

report_dft_drc_violations

Reports DFT DRC violations for the current design.

SYNTAX

string **report_dft_drc_violations**
[-test_mode test_mode_label]
[-rule *rule_list*]
[-summary]

Data Types

test_mode_label string
rule_list list

ARGUMENTS

-test_mode

Specifies the mode in which we are reporting DFT DRC violations. Argument examples are Internal_scan and ScanCompression_mode.

-rule

Specifies the collection of rule or rule type or rule category for which the DFT DRC violations must be reported.

-summary

Show a summary of the violations at the end of the report.

DESCRIPTION

This command reports DFT DRC violations on the current design against the test design rules specified by the **report_drc_violations -rule** for the mode specified by the **report_drc_violations -test_mode**.

EXAMPLES

The following example displays DFT DRC violations report on the current design aganist the specified rule category for the specified

test mode.

```
prompt> report_dft_drc_violations -rule B -test_mode all_dft
--------------------------------------------------------------------------------
 Report : DFT DRC Violations
     Mode   : all_dft
--------------------------------------------------------------------------------
 Warning: Unconnected module internal net ( des_decrypt/n1459 ). (B10-1)
 Warning: Unconnected module internal net ( des_decrypt/n1458 ). (B10-2)
 Warning: Undriven instance input pin ( /des_c_0/state_reg/CP ). (B12-1)
 Warning: Unconnected module input pin ( des_decrypt/i_key[64] ). (B8-1)
 Warning: Unconnected module input pin ( des_encrypt/i_key[64] ). (B8-2)
 Warning: Undriven module internal net ( des_decrypt/round[0] ). (B9-1)
```

The following example displays DFT DRC violations report on the current design aganist the specified rule type for the specified test mode.

```
prompt> report_dft_drc_violations -rule B10 -test_mode all_dft
--------------------------------------------------------------------------------
 Report : DFT DRC Violations
     Mode   : all_dft
--------------------------------------------------------------------------------
 Warning: Unconnected module internal net ( des_decrypt/n1459 ). (B10-1)
 Warning: Unconnected module internal net ( des_decrypt/n1458 ). (B10-2)
```

The following example displays DFT DRC violations report on the current design aganist the specified rule for the specified test mode.

```
prompt> report_dft_drc_violations -rule B10-2 -test_mode all_dft
--------------------------------------------------------------------------------
 Report : DFT DRC Violations
     Mode   : all_dft
--------------------------------------------------------------------------------
 Warning: Unconnected module internal net ( des_decrypt/n1458 ). (B10-2)
```

SEE ALSO

dft_drc(2)
current_design(2)

report_dft_insertion_configuration

Displays options set by the **set_dft_insertion_configuration** command.

SYNTAX

status **report_dft_insertion_configuration**

ARGUMENTS

The **report_dft_insertion_configuration** command has no arguments.

DESCRIPTION

This command displays options set by the **set_dft_insertion_configuration** command on the current design.

EXAMPLES

The following is an example of the **report_dft_insertion_configuration** command:

```
prompt> report_dft_insertion_configuration
****************************************
Report : Dft Insertion Configuration
Design : top
Version: P-2019.03-SP4
Date   : Mon Aug 26 06:55:25 2019
****************************************

Options                    Status
-------                  ------
UnScan                     False
1
```

SEE ALSO

set_dft_insertion_configuration(2)

report_dft_isolation

Report the UPF isolation strategies specified by set_dft_isolation command.

SYNTAX

status **report_dft_isolation**
-ref_domain *power_domain*
[-dft_target_domain *target_power_domain*]
[-type *type_list*]

Data Types

power_domain string
target_power_domain *string*
type_list list

ARGUMENTS

-ref_domain *power_domain*

Specify the domain where **-ref_strategy** is defined.

-dft_target_domain *target_power_domain*

Specify the target power domain of the connection.

-type *type_list*

Specify the connection types.

DESCRIPTION

This command reports the strategies that were specified by set_dft_isolation commands.

EXAMPLES

The following example reports isolation strategies on domain pd1 for type scan_in.

```
prompt> report_dft_isolation iso1 -ref_domain pd1 -type scan_in
iso1 : pd1(domain), scan_in(type)
```

SEE ALSO

set_isolation(2)
set_dft_isolation(2)

report_dft_location

Reports the DFT hierarchy location specifications for the current design.

SYNTAX

status **report_dft_location**

ARGUMENTS

The **report_dft_location** command has no arguments.

DESCRIPTION

The **report_dft_location** command is used to report the DFT hierarchy location specification that have been applied to the current design by the **set_dft_location** command.

User-defined locations are reported. Default locations are not reported.

EXAMPLES

The following example shows how two user-defined DFT locations are reported:

```
prompt> set_dft_location -include {CODEC} UDFTSTUFF
Accepted DFT location specification
1
prompt> set_dft_location -include {PLL} UCLKGEN
Accepted DFT location specification
1
prompt> report_dft_location
Design Name      DFT Partition Name  DFT Type        DFT Hierarchy Location
_____________    ___________________ _________    ____________________
top              default_partition   CODEC           UDFTSTUFF
top              default_partition   PLL             UCLKGEN
1
```

SEE ALSO

insert_dft(2)
set_dft_location(2)

report_dft_signal

Displays options specified by the **set_dft_signal** command.

SYNTAX

status **report_dft_signal**

ARGUMENTS

The **report_dft_signal** command has no arguments.

EXAMPLES

The following command reports the DFT signals specified for the current design:

```
prompt> report_dft_signal
****************************************
Report : DFT signals
Design : top
Version: P-2019.03
Date   : Thu Feb  7 06:53:06 2019
****************************************

----------------------------------------
TEST MODE: all_dft
VIEW    : Specification
----------------------------------------
Port    SignalType      Active  Hookup  Usage  Partition
------- ----------      ------ ------ ----- ------
SE      ScanEnable      -      -      Scan   default_partition
wrp_clk wrp_clock       -      -             default_partition
si0     ScanDataIn      -      -      -      default_partition
si1     ScanDataIn      -      -      -      default_partition
si2     ScanDataIn      -      -      -      default_partition
so0     ScanDataOut     -      -      -      default_partition
so1     ScanDataOut     -      -      -      default_partition
so2     ScanDataOut     -      -      -      default_partition

----------------------------------------
```

```
TEST MODE: all_dft
VIEW    : Existing DFT
-----------------------------------------
Port    SignalType       Active  Hookup  Usage  Timing
-------  ----------       ------  ------  -----  ------
CLK     MasterClock      1       -       -      P 100.0 R 45.0 F 55.0
iso1    Constant         0       -       -
iso2    Constant         0       -       -
wrp_clk  MasterClock     1       -       -      P 100.0 R 45.0 F 55.0
```

SEE ALSO

set_dft_signal(2)

report_disable_tie_insert

Reports all the tie pins which are marked by set_disable_tie_insert.

SYNTAX

status **report_disable_tie_insert**

DESCRIPTION

List all the disabled tie insert pins on which tie cell will not be inserted.

EXAMPLES

prompt>set_disable_tie_insert -objects [get_pins {h1/reg_2_/SE h1/reg_2_/SI}]

prompt>report_disable_tie_insert

Report:Disabled tie ports and pins

---

h1/reg_2_/SE

h1/reg_2_/SI

SEE ALSO

add_tie_cells(2)
set_disable_tie_insert(2)
reset_disable_tie_insert(2)

report_disable_timing

Reports disabled timing arcs in the current design.

SYNTAX

string **report_disable_timing**
[-mode *mode*]
[-nosplit]
[*object_spec*]

Data Types

mode collection
object_spec collection

ARGUMENTS

-mode *mode*

A single mode to be used for the report. This can be a collection or the name of a valid mode. If not specified, the current mode will be used.

-nosplit

Prevents line splitting and facilitates writing scripts to extract information from the report output. If you do not use this option, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line starting in the correct column.

object_spec

Limits disabled arc reporting to the specified list of cells or ports. Provide the list or collection of cells or ports as an argument to the command.

DESCRIPTION

Reports disabled timing arcs in the current design. Timing arcs can be disabled in several ways. You can disable a timing arc by using the **set_disable_timing** command. The following symbols are used to explain why a timing arc is disabled:

c : The propagation of case-analysis values

C : The presence of conditional arcs (arcs that have a when statement defined in the library)

d : The presence of default arcs (when the **time.disable_cond_default_arcs** application option is set to **true**)

f : Disabling of false net-arcs

l : Loop breaking

L : db inherited loop breaking

p : The propagation of constant values

u : Arcs disabled by using the **set_disable_timing** command.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-mode** option.

EXAMPLES

The following example shows that the timing arc of a cell named *U1/U2* from pin *A* to pin *Z* is disabled.

```
prompt> set_disable_timing {n2_i} -from A -to Z
prompt> report_disable_timing

****************************************
Report : disable_timing
Module : counter
Mode   : m2
Version:
Date   : Thu Jan  5 19:39:17 2017
****************************************

Flags :    c  case-analysis
           C  Conditional arc
           d  default conditional arc
           f  false net-arc
           l  loop breaking
           L  db inherited loop breaking
           p  propagated constant
           u  user-defined

Cell or Port  From  To  Sense         Flag  Reason
-------------------------------------------------------------
n2_i          A     Z   negative_unate  u

1
```

The following example specifies that the timing arc of cell *ff4* from pin *CP* to pin *TE* and *TI* are disabled as a result of a case-analysis constant of *0* propagated to pin *TE* of cell *ff4*. Note that the actual case value is set on another pin *A* and the propagation path to *ff4/TE* is inverting.

```
prompt> set_case_analysis 0 {p_in in0}
prompt> report_disable_timing

****************************************
Report : disable_timing
```

```
Module : counter
Mode   : m2
Version:
Date   : Thu Jan  5 19:39:17 2017
****************************************

Flags :    c  case-analysis
           C  Conditional arc
           d  default conditional arc
           f  false net-arc
           l  loop breaking
           L  db inherited loop breaking
           p  propagated constant
           u  user-defined

Cell or Port  From  To  Sense          Flag  Reason
-------------------------------------------------------------
o_reg4        CP    D   hold_clk_rise   c     D = 0
o_reg4        CP    D   setup_clk_rise  c     D = 0
o_reg4        CP    CD  recovery_rise_clk_rise
                            c     D = 0
p_out                       c     p_out = 1

1
```

The following example provides a list of the required cells as an argument to the command to view disabled arcs in specific cells. In the previous example, to view disabled arcs for *o_reg4* instance, do the following:

```
prompt> report_disable_timing  [get_cells { o_reg4 }]

****************************************
Report : disable_timing
Module : counter
Mode   : m2
Version:
Date   : Thu Jan  5 19:39:17 2017
****************************************

Flags :    c  case-analysis
           C  Conditional arc
           d  default conditional arc
           f  false net-arc
           l  loop breaking
           L  db inherited loop breaking
           p  propagated constant
           u  user-defined

Cell or Port  From  To  Sense          Flag  Reason
-------------------------------------------------------------
o_reg4        CP    D   hold_clk_rise   c     D = 0
o_reg4        CP    D   setup_clk_rise  c     D = 0
o_reg4        CP    CD  recovery_rise_clk_rise
                            c     D = 0

1
```

SEE ALSO

remove_disable_timing(2)
set_disable_timing(2)

report_dont_touch

Reports dont_touch information for design objects.

SYNTAX

string **report_dont_touch**
  [-all]
  [*cell_net_list*]

list *cell_net_list*

ARGUMENTS

-all

Reports for all dont_touch cells, and nets.

cell_net_list

Lists cells or nets to report. If you do not specify this or the **-all** option, a summary report will be printed, listing the total number of dont_touch cells and nets.

DESCRIPTION

Displays dont_touch information for cells and nets in the current design. For each dont_touch object, the reasons for the dont_touch status will be listed.

EXAMPLES

The following example shows a summary report:

```
prompt> report_dont_touch

Report : dont_touch
0 of 251 instances are dont_touch
4 of 275 nets are dont_touch
1
```

The following example shows a report for one cell that has had **set_dont_touch** applied to it:

```
prompt> report_dont_touch cell_3/ABO_CELL_19

Report : dont_touch

Instance cell_3/ABO_CELL_19 (ND2D1BWP)
 user
1 of 1 instances are dont_touch
0 of 0 instances are dont_touch
```

The following example lists all dont_touch objects and the reasons for the dont_touch status:

```
prompt> report_dont_touch -all

Report : dont_touch

Instance cell_3/ABO_CELL_19 (ND2D1BWP)
 user

1 of 251 instances are dont_touch
0 of 275 nets are dont_touch
1
```

SEE ALSO

set_dont_touch(2)
report_cell(2)

report_drc_errors

Writes out an error report of physical DRC errors from the specified error data. The **report_drc_error** command supports different reporting styles.

SYNTAX

status **report_drc_error**
-error_data *drc_error_data*
[-error_type *drc_error_types*]
[-layers *drc_error_layers*]
[-ignore_type *drc_error_types*]
[-ignore_layers *drc_error_layers*]
[-of_objects *drc_error_objects*]
[-report_type error_type | error_layer | detailed | matrix]
[-nosplit]

Data Types

drc_error_data collection
drc_error_types collection
drc_error_layers collection or string
drc_error_objects collection

ARGUMENTS

-error_data *drc_error_data*

Specifies the error data for finding objects.

-error_type *drc_error_types*

Reports errors within the specified *drc_error_types*. An error is included in the output if it matches any one of the *drc_error_types*. Use the **get_drc_error_types** command to specify *drc_error_types*.

-layers *drc_error_layers*

Reports errors within the specified *drc_error_layers*. An error is included in the output if it matches any one of the *drc_error_layers*. The command accepts strings or **[get_layers]** result as an argument to the **-layers** option.

-ignore_type *drc_error_types*

Reports errors exclude the specified *drc_error_types*. An error is excluded in the output if it matches any one of the *drc_error_types*. Use the **get_drc_error_types** command to specify *drc_error_types*.

-ignore_layers *drc_error_layers*

Reports errors exclude the specified *drc_error_layers*. An error is excluded in the output if it matches any one of the *drc_error_layers*. The command accepts strings or **[get_layers]** result as an argument to the **-layers** option.

-of_objects *drc_error_objects*

Reports errors with the specified *drc_error_objects*. In this case, the *drc_error_objects* are error instances. Use the **get_drc_errors** command to specify *drc_error_objects*.

-report_type error_type | error_layer | detailed | matrix

Writes out the specified report type. Three types are supported: *error_type*, *error_layer*, *detailed* and *matrix*. You can specify only one report type. By default, if no *report_type* is specified, the report type is *error_type*.

-nosplit

Write out the report without splitting long lines. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line splitting and facilitates writing software to extract information from the report output.

DESCRIPTION

The **report_drc_error** command prints a collection or summary of physical DRC errors in the specified error data that match certain criteria. The command returns true if any physical DRC errors match the criteria and successfully reported. If no objects match the criteria, the error status will be reported.

There are four types of report supported by the command: *error_type*, *error_layer*, *detailed* and \fmatrix. Use the **-report_type** option to specify the report type. The *error_type* report type writes out a summary of error information in type order. The *error_layer* report type writes out a summary of error information in layer order. The *detailed* report writes out each error instance information one-by-one. The information includes the ID, type, layer, number of errors in this type, description, status and bbox. The *matrix* report writes out a table format for all the error information.

The **-error_type** option filters the errors by the specified types. You can specify more than one **error_type** here by enclosing each error type in curly braces: *{{$errType1} {$errType2}}*

The **-layers** option filters the errors by the specified layers. You can specify more than one layer by enclosing each layer in curly braces: *{{$errLayer1} {$errLayer2}}*

The **-ignore_type** option filters out the errors by the specified types. You can specify more than one **error_type** here by enclosing each error type in curly braces: *{{$errType1} {$errType2}}*

The **-ignore_layers** option filters out the errors by the specified layers. You can specify more than one layer by enclosing each layer in curly braces: *{{$errLayer1} {$errLayer2}}*

The **-of_objects** option filters the errors by the specified error instances. Use the **get_drc_errors** command to specify the instances.

You can specify one or more filtering options to customize the report. If you specify *"Short"* and *"Net"* error type with *"METAL"* and *"METAL2"* layers, you can get errors with *"Short"* and *"METAL"*, with *"Short"* and *"METAL2"*, with *"Net"* and *"METAL"*, with *"Net"* and *"METAL2"*. If you specify the **-of_objects** option, the command writes out only the errors for the specified objects.

For long reports you do not want to write to the console, use the **redirect** command to write the output to a file.

EXAMPLES

The following example displays an error_type report of the errors with error type 'Net'.

```
prompt> set type [get_drc_error_types -error_data $data {Net}]
prompt> report_drc_error -error_data $data -error_type $type -report_type error_type

****************************************
Report : Report drc errors
Design : test
Data   : zroute.err
Type   : error_type
Version: O-2018.06-SP2-BETA
Date   : Mon Jul  9 15:26:16 2018
****************************************

ErrorSet                          Total   Visible    Ignored Fixed
--------------------------------------------------------------------------------
Design : test                        54     3         0      0
    Data : zroute.err                  54     3         0      0
        Net                      3     3         0      0
          METAL2 (18)                1     1         0      0
          METAL3 (28)                1     1         0      0
          METAL4 (31)                1     1         0      0
1
```

The following example displays an error_layer report of the errors with error layer 'METAL4'.

```
prompt> report_drc_error -error_data $data -layers {METAL4} -report_type error_layer

****************************************
Report : Report drc errors
Design : test
Data   : zroute.err
Type   : error_layer
Version: O-2018.06-SP2-BETA
Date   : Mon Jul  9 15:28:28 2018
****************************************

ErrorSet                          Total   Visible    Ignored Fixed
--------------------------------------------------------------------------------
Design : test                        54     5         0      0
    Data : zroute.err                  54     5         0      0
        METAL3 (28)-METAL4 (31)          2     2         0      0
          Open Locator                 1     1         0      0
          Route Spacing                1     1         0      0
        METAL4 (31)                    2     2         0      0
          Net                    1     1         0      0
          Route Short                  1     1         0      0
        METAL5 (3)-METAL4 (31)           1     1         0      0
          Route Spacing                1     1         0      0
1
```

The following example displays a detailed report of the errors with error type 'Net' and 'Route Spacing' for layer 'METAL3':

```
prompt> set data [open_drc_error_data -file_name zroute.err]
prompt> set type [get_drc_error_types -error_data $data {{Route Spacing} {Net}}]
prompt> report_drc_error -error_data $data -error_type $type \
  -layers {METAL3} -report_type detailed

****************************************
Report : Report drc errors
Design : test
Data   : zroute.err
Type   : detailed
Version: O-2018.06-SP2-BETA
Date   : Mon Jul  9 14:54:16 2018
****************************************

Error ID         : 3
Error Type       : Route Spacing
Error Layer      : METAL2 (18)-METAL3 (28)
Number of errors  : 6
Description      : Spacing violation between A0_9_ and B0_9_
                   on layers {METAL2 (18) METAL3 (28)} --
                   direction: upper left to lower right,
                   required: 1.2000 um, actual: 0.6000 um
Status           : error
Bbox             : {261.0000 301.8000} {261.4000 303.0000}

Error ID         : 49
Error Type       : Net
Error Layer      : METAL3 (28)
Number of errors  : 3
Description      : Error on nets { B0_5_ hmg_out44_3_} and layers {METAL3 (28)}
Status           : error
Bbox             : {237.8000 150.6000} {238.2000 151.8000}
1
```

The following example displays a matrix report of the errors:

```
prompt> set data [open_drc_error_data -file_name zroute.err]
prompt> report_drc_error -error_data $data -report_type matrix

****************************************
Report : Report drc errors
Design : ORCA
Data   : zroute.err
Type   : matrix
Version: R-2020.09-DEV
Date   : Fri Nov  8 17:39:39 2019
****************************************

                    | METAL  METAL2  METAL3  METAL4  VIA  | TOTALS BY TYPE
-------------------------------------------------------------------------------------
Diff net spacing        | 18     15      2       -      -   | 35
Less than minimum area  | -      4       -       -      -   | 4
Less than minimum width | -      -       2       -      -   | 2
Needs fat contact       | -      8       -       -      -   | 8
Same net spacing        | 6      -       -       -      -   | 6
Same net via-cut spacing| -      -       -       -      2   | 2
```

```
Short              | 110   226    129    17     -   | 482
-------------------------------------------------------------------------------
                   | METAL  METAL2  METAL3  METAL4  VIA  | 539
TOTALS BY LAYER    | 134   253    133    17     2    |
1
```

SEE ALSO

redirect(2)
get_drc_errors(2)
get_drc_error_types(2)
report_nets(2)
report_cells(2)

report_eco_bus_buffer_patterns

Reports information for the ECO bus buffer patterns in the current design.

SYNTAX

report_eco_bus_buffer_patterns
[*patterns*]

Data Types

patterns list

ARGUMENTS

patterns

Lists ECO bus buffer patterns to report. If you do not specify this option, the command reports all ECO bus buffer patterns.

DESCRIPTION

Reports information about ECO bus buffer patterns in the current design.

EXAMPLES

The following example reports all ECO bus buffer patterns in the current design.

```
prompt> report_eco_bus_buffer_patterns
****************************************
Report : Bus Buffer Pattern Report
Design : b6c
Version:
Date   :
****************************************

============================================================================
Patterns with Constant Distance between Neighbor Bits
============================================================================
```

```
Name              First Buffer    Measure Side  Distance    Repeat After
------------------------------------------------- ------------------------
phbl_2_3          bottom          left          2.00        3
phbr_0_2_looooooooooooooooongName
                  bottom          right         0.00        2
phbr_1p5_b0       bottom          right         1.51        No Repeat
------------------------------------------------- ------------------------
* Distance is between measure sides of two buffers of adjacent bits,
  unit is 1.00um

============================================================================
Patterns with User Specified Distance between Neighbor Bits
============================================================================

Name              First Buffer    Measure Side    Distances
----------------------------------------------------------------------------
pvrt_v1           right           top             5.0 -1.1
----------------------------------------------------------------------------
* Distances are between measure sides of two buffers of adjacent bits,
  unit is 1.00um
```

SEE ALSO

create_eco_bus_buffer_pattern(2)
get_eco_bus_buffer_patterns(2)
remove_eco_bus_buffer_patterns(2)

report_eco_physical_changes

Reports the cell displacements and individual net length changes resulting from the **size_cell**, **add_buffer**, and **add_buffer_on_route** commands.

SYNTAX

status **report_eco_physical_changes**
[-type *type_list*]
[-min_displacement *min_displacement_value*]
[-min_estimated_displacement *min_estimated_displacement_value*]
[-min_net_length_ratio *min_net_length_ratio_value*]
[-min_estimated_length *min_estimated_length_value*]
[-cells *specified_cells*]
[-ignore_filler_references *references*]
[-min_estimated_routing_length_ratio *ratio*]
[-min_optimal_routing_length *min_optimal_routing_length_value*]
[-min_cluster_cell_distance *min_cluster_cell_distance_value*]
[-min_cell_number_in_cluster *min_cell_number_in_cluster_value*]
[-min_displacement_threshold_for_cluster *min_displacement_threshold_for_cluster_value*]
[-max_displacement_threshold_for_cluster *max_displacement_threshold_for_cluster_value*]

Data Types

type_list	list
min_displacement_value	float
min_estimated_displacement_value	float
min_net_length_ratio_value	float
min_estimated_length_value	list
specified_cells	list or collection
references	list or collection
ratio	float
min_optimal_routing_length_value	float
min_cluster_cell_distance_value	float
min_cell_number_in_cluster_value	int
min_displacement_threshold_for_cluster_value	float
max_displacement_threshold_for_cluster_value	float

ARGUMENTS

-type *type_list*

Specifies the types of additional data reported, aside from a global summary. Without this option, the command reports only the

global summary.

The valid values are one of "**cell_displacement**," "**estimated_cell_displacement**," "**net_length**," "**cluster**," and "**all**."

Specify "**cell_displacement**" to report the cell displacement. This report shows the original cell location, the current cell location, and the displacement. The original location is where cell is inserted or sized, but if the cell is placed by the **place_eco_cells** command, the original location is placement location before legalization. An asterisk after the displacement means that the cell is out of its connection boundary box.

Specify "**estimated_cell_displacement**" to identify ECO cells (newly added buffers, inverter pairs, sized cells) that would be ineffective to fix timing QoR and could have negative impact of timing QoR on non-ECO part of the design. To identify the targeted ECO cells to be reverted, we use placement resource feasibility analysis (PRFA) to check if an ECO cell could have placement resource legalized within a certain threshold specified by the **-min_estimated_displacement** option.

The reverting candidate ECO cells are ECO cells that could only have resource greater than or equal to the given threshold. The tool creates a collection called eco_estimated_large_displacement_cells for such ECO cells. For ECO cells in collection above, report the current cell location, the estimated cell location, and the displacement to the nearest unoccupied placement area. And report estimated cell displacement summary. The summary is irrelevant to the threshold.

It does not guarantee the location will be legal, so the prediction of placement is very optimistic. If the unoccupied placement area legalization were performed, the real displacement could be larger than the predicted displacement. User can decide to use **revert_eco_changes** to revert ECO cells with large displacement.

Specify "**net_length**" to report the lengths of individual nets affected by ECO changes, including the estimated route length, actual route length, and ratio of real to estimated route length.

Specify "**cluster**" to report cell clusters. The report shows cluster id and number of cells in this cluster.

Specify "**all**" to generate both the cell displacement and net length reports.

-min_displacement ***min_displacement_value***

Filters out cells whose displacements are less than a specified amount, in microns. This can be used only when "**cell_displacement**" or "**all**" is specified in the **-type** option. This option does not affect the global summary. The default is 5.0 times of site height.

-min_estimated_displacement ***min_estimated_displacement_value***

Specify a user-defined displacement threshold for estimated cell displacement. The option can only be used when "estimated_cell_displacement" or "all" is specified in the **-type** option. The default is infinity. If not specified, the collection eco_estimated_large_displacement_cells is empty but the estimated cell displacement summary is reported. This option does not affect the global summary and estimated cell displacement.

-min_net_length_ratio ***min_net_length_ratio_value***

Filters out individual nets whose length ratios are less than a specified number. This can be used only when "**net_length**" or "**all**" is specified in the **-type** option. This option does not affect the global summary. The default is 1.1, corresponding to a real-to-estimated length ratio of 110 percent.

-min_estimated_length ***min_estimated_length_value***

Filters out individual nets whose estimated lengths are less than a specified number of microns. This can be used only when "**net_length**" or "**all**" is specified in the **-type** option. It works together with the **-min_net_length_ratio** option to control the reporting of individual net lengths. This option does not affect the global summary.

-cells ***specified_cells***

Reports the cell displacement and individual net lengths only for the specified cells. This option can be used together with the other options.

-ignore_filler_references ***references***

Specify the references of filler cells which can be ignored when calculating estimated cell location by PRFA. Unfixed filler cells of the specified references will be ignored and the site spaces occupied by them are treated as free spaces by PRFA. Other filler cells that not of the specified references are still treated as normal cells and cannot not be ignored. The option can only be used when "estimated_cell_displacement" or "all" is specified in the **-type** option.

-min_estimated_routing_length_ratio ***ratio***

Specify the threshold for ratio of estimated routing length to optimal one, 1 by default if no specified. This options helps to predict and report ECO cells with estimated location out of driver-load bounding box. When specified, collect ECO cells which ratio is greater than the specified ratio. The tool creates a collection called eco_estimated_cells_outside_driver_load_bbox for such ECO cells. The option can only be used when "estimated_cell_displacement" or "all" is specified in the **-type** option.

-min_optimal_routing_length ***min_optimal_routing_length_value***

Specify the minimal optimal routing length from driver to load. The option can only be used with the **-min_estimated_routing_length_ratio** option. When specified, only if the optimal routing length for an ECO cell from its driver to load is greater than the specified minimal optimal routing length, the tool calculates the estimated routing length ratio and collects ECO cells which ratio is greater than the specified ratio as option **-min_estimated_routing_length_ratio** described. Otherwise, collect ECO cells which ratio is greater than the specified ratio. This option is to help you filter out the ECO cell which drive and load are too close to make the estimated routing length ratio large but actually the ECO cell is not far away from its driver-load bounding box.

-min_cluster_cell_distance ***min_cluster_cell_distance_value***

Specify min distance between cells in a cluster, this option can be only used when "cluster" is specified **-type**.

-min_cell_number_in_cluster ***min_cell_number_in_cluster_value***

Specify minimal number of cells in a cluster to report, this option can be only used when "cluster" is specified **-type**. The default value for this option is 2.

-min_displacement_threshold_for_cluster ***min_displacement_threshold_for_cluster_value***

Specify minimal displacement threshold for each cell in cluster, this option can be only used when "cluster" is specified **-type**. Number of cells in a cluster with displacement over this threshold will be reported.

-max_displacement_threshold_for_cluster ***max_displacement_threshold_for_cluster_value***

Specify maximal displacement threshold for each cell in cluster, this option can be only used when "cluster" is specified **-type**. Number of cells in a cluster with displacement over this threshold will be reported.

DESCRIPTION

This command reports the physical changes to the current design or to specified cells after an ECO change list has been applied. The generated report includes a global summary, and optionally, the individual cell displacements and changed net lengths.

In a report of cell displacements, the individual cells are sorted by amount of displacement, in descending order, in two groups: first the cells moved out of their connection boundary box (these are marked with an asterisk), followed by cells that remain inside their connection boundary box.

In a report of changed net lengths, the individual nets are sorted by real-to-estimated length ratios in descending order.

The following editing commands are supported in the change list. Changes caused by other ECO editing commands are not reported.

add_buffer
size_cell

add_buffer_on_route

It is recommended that you generate the ECO report after ECO legalization or ECO routing.

EXAMPLES

The following example generates a global summary report.

```
prompt> report_eco_physical_changes
****************************
Report : eco physical change
 ...

****************************
Summary
========
Cell Displacement
------------------------------------------------------------
Total eco cell number                  : 430
Max displacement                       : 20 micron
Avg  displacement                      : 5 micron
Cells outside connection bbox          : 10

eco net route length vs estimated length ratio
------------------------------------------------------------
Total eco net number                   : 400
Max ratio                              : 150%
Avg ratio                              : 120%
```

The following example generates a cell displacement report.

```
prompt> report_eco_physical_changes -type cell_displacement -min_displacement 5

(The global summary section is shown in the previous example)

Report cell location legalization displacement (meaningful after eco legalization):
============================================================
cell name      before legalization   after legalization  displacement(micron)
------------------------------------------------------------------------------
eco_cell_0     (82.64, 98.32)       (82.80, 108.96)     10.80 *
eco_cell       (99.14, 92.58)       (102.60, 97.76)     8.64
------------------------------------------------------------------------------
Total 2 cells reported.
Max displacement: 10.80 micron
Avg displacement: 9.74 micron
Min displacement threshold: 5.00 micron
Cells outside connection bbox (Marked by *): 1
```

The following example generates a report of individual net lengths.

```
prompt> report_eco_physical_changes -type net_length \
  -min_net_length_ratio 1.02 -min_estimated_length 10.23

(The global summary section is shown in the first example)
```

```
Report individual net index and name:
==============================

index                    name
-------------------------------------------------------
net_1                    UPC_DATA[2]
net_2                    eco_net
net_3                    UPC_DATA[10]
net_4                    eco_net_0
-------------------------------------------------------
Total 4 individual nets reported.

Report individual net length (meaningful after eco route):
==========================================

index      estimated(micron)    real(micron)      ratio(real/estimated)
-------------------------------------------------------------------------------
net_3      10.52                19.22             183.34%
net_1      30.09                36.27             121.50%
net_2      56.10                65.48             117.98%
net_4      34.22                35.21             103.31%
-------------------------------------------------------------------------------
Total 4 individual nets reported.
Max ratio: 183.34%
Avg ratio: 131.53%
Min ratio threshold: 102%
Min estimated length threshold: 10.23 micron
```

The following example reports estimated cell displacement.

```
prompt> set references [get_lib_cells {lib1/FILL1 lib1/FILL2}]
prompt> report_eco_physical_changes \
  -type estimated_cell_displacement -min_estimated_displacement 1 \
  -ignore_filler_references $references

Report estimated cell displacement to nearest free site:
=============================================================================================
cell name     current location       estimated location      displacement(micron)
---------------------------------------------------------------------------------
mult_26/U4751  (24.82, 21.94)        (32.88, 31.20)        12.28
mult_26/U4754  (23.08, 23.09)        (33.69, 23.14)        10.61
mult_26/U4753  (23.11, 21.36)        (30.27, 14.50)        9.92
mult_26/U4755  (24.57, 22.95)        (33.69, 23.14)        9.12
mult_26/U4752  (22.09, 23.23)        (27.21, 21.98)        5.27
---------------------------------------------------------------------------------
Total 5 eco cells may have large displacement (threshold 1.00 micron).
Note: You can use revert_eco_changes -cells to revert eco cells with large displacement.
Max displacement:                12.28 micron
Avg displacement:                9.44 micron
Std deviation:                   2.33 micron
Number of cell moved (estimated):          5 cells (out of 5 cells)

prompt> get_cells $eco_estimated_large_displacement_cells
```

The following example reports estimated cell displacement and collects ECO cells with estimated location outside the driver-load bounding box.

```
prompt> report_eco_physical_changes \
  -type estimated_cell_displacement -min_estimated_displacement 1 \
  -min_estimated_routing_length_ratio 5 \
  -min_optimal_routing_length 2.5
prompt> get_cells $eco_estimated_cells_outside_driver_load_bbox
```

The following example reports cell clusters.

```
prompt> report_eco_physical_changes \
  -type cluster -min_displacement_threshold_for_cluster 1 \
  -min_cell_number_in_cluster 5 \
  -min_cluster_cell_distance 2
```

```
Report cell cluster detail information:
========================================
eco_cluster_id        number of cells       number of cells > min displacement
                                            number of cells > max displacement
-------------------------------------------------------------------------------------------------------
eco_cluster_id 0      10                    3                    0
eco_cluster_id 1      6                     0                    0
eco_cluster_id 2      5                     0                    0
-------------------------------------------------------------------------------------------------------
Total number of clusters is 3, minimum number of cells in a cluster is 5,
minimum distance between two cells in a cluster is 2.
Note: You can use command like: get_cells -hier -filter "eco_cluster_id==1" to
get cell collection with attribute eco_cluster_id = 1.
```

SEE ALSO

add_buffer(2)
add_buffer_on_route(2)
report_cell_feasible_space(2)
revert_eco_changes(2)
size_cell(2)

report_eco_placement_net_weight

Report user specified net weight.

SYNTAX

status **report_eco_placement_net_weight**
[*-nets collection_of_net*]
[*-output file_name*]

Data Types

collection_of_net list or collection
file_name string

ARGUMENTS

-nets *collection_of_net*

Report net weight for given nets. The option *-nets* can not be used with *-output*. They are mutually exclusive. *Output format: "$n1 $w1".* If no option is used, the command reports all net weight setting.

-output *file_name*

Write out the internal net weight mapping table with the format below. User can load it when needed. *Output format: "set_eco_placement_net_weight -nets $n1 -weight $w1".*

DESCRIPTION

This command report user specified net weight setting. Please refer to the man page of place_eco_cells for the flow usage about user-defined net weight based coarse placement.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports all net weight setting.

```
prompt> report_eco_placement_net_weight
```

The following example reports specified net weight setting.

```
prompt> report_eco_placement_net_weight \
    -nets $netList
```

The following example writes out net weight setting with tcl format.

```
prompt> report_eco_placement_net_weight -output $file_name
```

SEE ALSO

place_eco_cells(2)
set_eco_placement_net_weight(2)

report_edit_groups

report_edit_groups

Reports edit groups in the current design.

SYNTAX

int **report_edit_groups**
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*edit_group_list*]

Data Types

digits int
edit_group_list list

ARGUMENTS

-verbose

Displays verbose edit group information.

-nosplit

Retains long lines and does not split lines if column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13. The default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option to override the default.

edit_group_list

Specifies a list of edit groups to report. The list can contain edit group names, patterns, or collections. A collection can be specified by using the **get_edit_groups** command.

DESCRIPTION

The **report_edit_groups** command generates a report about the edit groups in the design. The report includes the edit group name and a description of the edit group. If **-verbose** is specified, the report also includes additional attributes and a list of objects contained in the edit group.

If *edit_group_list* is specified, those edit groups are reported. If it is not specified, then all edit groups in the design are reported.

EXAMPLES

The following example reports all edit groups in the design.

```
prompt> report_edit_groups *
Edit Group        Description
---------------------------------------------------------------------------
EDIT_GROUP_1      Contains 5 objects   ungroup_on_remove: never
EDIT_GROUP_2      Contains 1 object    ungroup_on_remove: last
EDIT_GROUP_3      Contains 8 objects   ungroup_on_remove: second-to-last
1
```

The following example reports verbose information for the edit group named "EDIT_GROUP_1".

```
prompt> report_edit_groups -verbose EDIT_GROUP_1
Edit Group        Description
---------------------------------------------------------------------------
EDIT_GROUP_1      Contains 5 objects   ungroup_on_remove: never
            status:       unplaced
            in_edit_group:  false
            can_transform:  true
            objects:      blkg
                        I_ORCA_TOP/I_PARSER/U176
                        d0
                        mypgr
                        EDIT_GROUP_3

1
```

SEE ALSO

add_to_edit_group(2)
create_edit_group(2)
get_edit_groups(2)
remove_edit_groups(2)
remove_from_edit_group(2)

shell.common.report_default_significant_digits(3)

report_editability

Reports the hierarchical edit control of specified blocks.

SYNTAX

int **report_editability**
[-blocks *list_of_blocks*]
[-nosplit]

Data Types

list_of_blocks	string or collection

ARGUMENTS

-blocks *list_of_blocks*

The hierarchical edit control of specified blocks would be reported with respect to the current top-level context. If this option is not specified, the default behavior shall report the hierarchical editability of only the current_block.

-nosplit

Retains long lines and does not split lines if column overflows.

DESCRIPTION

This command reports the true/false value of hierarchical edit controls corresponding to each of the specified blocks. A true value means the block is editable and false value means the block is uneditable.

Returns an integer indicating number of reported blocks with hierarchical edit controls.

EXAMPLES

The following example reports the hierarchical edit control of a single block:

```
prompt> report_editability -blocks blk1
****************************************
Report : report_editability
```

```
Top Design: lib:top.design
Version: M-2016.12-SP4-BETA
Date   : Wed Feb  8 06:20:36 2017
****************************************
Block      Is_editable
--------------------------------------------------------------------------------
blk1       false
1
```

The following example reports the hierarchical edit control of multiple blocks:

```
prompt> report_editability -blocks [get_blocks {blk1 blk2 blk3}]
****************************************
Report : report_editability
Top Design: lib:top.design
Version: M-2016.12-SP4-BETA
Date   : Wed Feb  8 06:20:36 2017
****************************************
Block      Is_editable
--------------------------------------------------------------------------------
blk1       false
blk2       true
blk3       false
3
```

SEE ALSO

get_editability(2)
set_editability(2)

report_ems_database

Reports the messages of an EMS database as text messages onto the standard output.

SYNTAX

bool **report_ems_database**
[-name *ems_database_name*]

Data Types

ems_database_name string

ARGUMENTS

-name *ems_database_name*

Specifies the name of the EMS database, whose contents are to be reported.

DESCRIPTION

This command reports the messages of an EMS database onto standard output. If option **-name** is not provided then the messages of the **current** EMS database are reported. If **current** EMS database does not exist, then the command errors out.

Option **-name** takes name of an EMS database as the argument. If the EMS database exists, then EMS messages present in it are reported out as text messages. The EMS database name specified, needs to obey EMS database naming convention, such as: EMS database should have .ems suffix. For example, cts.ems, test.ems etc.

EXAMPLES

The following example shows reporting ems messages from both current EMS database and also from an EMS database on disk. Then two files generated are compared to show that they are identical.

```
prompt> create_ems_database abc.ems
{abc.ems}
prompt> check_mv_design
1
prompt> report_ems_database > check_mv_design_curr_ems.txt
```

```
1
prompt> save_ems_database
prompt> close_ems_databases
prompt> report_ems_database -name abc.ems > check_mv_design_disk_ems.txt
prompt> sh diff -s check_mv_design_curr_ems.txt  check_mv_design_disk_ems.txt
Files check_mv_design_curr_ems.txt and check_mv_design_disk_ems.txt are identical.
```

SEE ALSO

create_ems_database(2)
close_ems_databases(2)
save_ems_database(2)

report_ems_rules

Reports information about user-defined EMS rules.

SYNTAX

report_ems_rules
[pattern]
[-all]

Data Types

pattern Collection of EMS rules

ARGUMENTS

pattern

This is an optional argument.It can be the name of a rule or a pattern capturing names of multiple rules or a collection of EMS rules. EMS rules by given name or matching given pattern or part of collection, should be already available in EMS memory; otherwise the command will fail with an error message. Option -all and argument "pattern" are mutually exclusive.

-all

This is an optional option. This is mutually exclusive with argument "pattern". This option reports information related to all user-defined rules in current EMS memory. This is also the default behavior of the command, when run without any options or arguments.

DESCRIPTION

Command **report_ems_rules** reports information related to one or more user-defined EMS rules. The information related to rule reported is name, severity, message and parameters.

EXAMPLES

```
prompt> create_ems_rule -name "TEMP-001" -severity "Info" -message "Pin:%pin has capacitance:%cap"
prompt> edit_ems_rule -name "TEMP-001" -parameters "name:pin type:string command:get_pins"
prompt> edit_ems_rule -name "TEMP-001" -parameters "name:cap type:double"
```

```
prompt> create_ems_rule -name "TMP-002" -severity "Error" -message "Cell %cell is not placed"
prompt> edit_ems_rule -name "TMP-002" -severity "Info"
prompt> edit_ems_rule -name "TMP-002" -message "The Cell %cell is not placed."
prompt> edit_ems_rule -name "TMP-002"-parameters "name:cell type:string command:get_cells"
prompt> report_ems_rules TEMP.*

Rule: TEMP-001
  Severity: Info
  Message: Pin:%pin has capacitance:%cap
  Parameter name: cap type: double command: NA
  Parameter name: pin type: string command: get_pins
prompt> report_ems_rules

Rule: TEMP-001
  Severity: Info
  Message: Pin:%pin has capacitance:%cap
  Parameter name: cap type: double command: NA
  Parameter name: pin type: string command: get_pins

Rule: TMP-002
  Severity: Info
  Message: The Cell %cell is not placed.
  Parameter name: cell type: string command: get_cells
```

SEE ALSO

create_ems_rule(2)
report_ems_rules(2)

report_essential_points

Reports the essential points that influence a specified set of objects.

SYNTAX

int **report_essential_points**
[*object_list*]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-list_scis]

Data Types

object_list list
mode_list list
corner_list list
scenario_list list

ARGUMENTS

-objects *object_list*

List of objects - pins/ports/nets/cells - for which to report the activity influencing essential points. For a net, we consider any essential point in its fanin. For pins/ports, we consider any essential point in the fanin; if the pin/port happens to be an essential point, we report/apply activity for the pin/port itself. For cells, we consider any essential point that is in the fanin of its input pins.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering on modes occurs.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If this option is not specified, no filtering occurs on corners.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Output of the command is generated separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options is specified, the command reports for the current scenario in the design.

-list_scis

Reports detailed information for each SCI (Special Control Input pins, such as clock pins, set/reset pins, scan enables, isolation cell control pins, power switch control pins etc) in the fanout of each selected essential point.

DESCRIPTION

This command takes a set of objects and for each specified scenario, generates a report of the essential points that influence these objects. To filter for scenarios, the *-modes*, *-corners* and *-scenarios* options can be used. The *-list_scis* option provides detailed information on the SCIs in the fanout of the reported essential points.

EXAMPLES

The following example shows the report generated by the **report_essential_points** command.

```
prompt> report_essential_points [get_nets {test_in1 clksel}]
Information: Activity for scenario scenario1 was cached, no propagation required. (POW-005)
****************************************
Report : report_essential_points
Design : ChipTop
Version: O-2018.06-SP2-BETA
Date   : Wed Jun 20 02:53:00 2018
****************************************
Inferring activity for the scenario: scenario1
Mode        : func
Corner      : max_corner
Time Unit   : 1ns
Fastest clock: clk2 with period 1.0
--------------------------------------------------------------------------------------------------------------
Essential Points for net 'test_in1' :
Object Type             : port
Object Name             : test_in1
Current Static Probability : 0.50
Current Toggle Rate       : 0.10
Receiver Type: scan_enable | Receiver Count: 10
--------------------------------------------------------------------------------------------------------------
Essential Points for net 'clksel' :
Object Type             : port
Object Name             : clksel
Current Static Probability : 0.50
Current Toggle Rate       : 0.10
Receiver Type: clock | Receiver Count: 9
Receiver Type: clock | Receiver Count: 1 (NEGATIVE)
--------------------------------------------------------------------------------------------------------------
```

In the following example, **report_essential_points** command is used with the *-list_scis* option to get a more detailed report with activity information for individual SCIs.

```
prompt> report_essential_points -list_scis [get_pins reg1/SE]
Information: Activity for scenario scenario1 was cached, no propagation required. (POW-005)
****************************************
```

```
Report : report_essential_points
       -list_scis
Design : ChipTop
Version: O-2018.06-SP2-BETA
Date   : Wed Jun 20 01:36:06 2018
****************************************
Inferring activity for the scenario: scenario1
Mode        : func
Corner      : max_corner
Time Unit   : 1ns
Fastest clock: clk2 with period 1.0
--------------------------------------------------------------------------------------------------------------------
Essential Points for pin 'reg1/SE' :
Object Type              : port
Object Name              : test_in1
Current Static Probability : 1
Current Toggle Rate      : 0
                                  Receiver        Current          Current        Current
Receiver                             Type    Activity Type  Static Probability    Toggle Rate

reg1/SE                          scan_enable      annotated              1            0
reg2/SE                          scan_enable      annotated              1            0
reg3/SE                          scan_enable      annotated              1            0
reg4/SE                          scan_enable      annotated              1            0
reg5/SE                          scan_enable      annotated              1            0
reg6/SE                          scan_enable      annotated              1            0
reg7/SE                          scan_enable      annotated              1            0
reg8/SE                          scan_enable      annotated              1            0
reg9/SE                          scan_enable      annotated              1            0
reg10/SE                          scan_enable       annotated              1            0
--------------------------------------------------------------------------------------------------------------------
```

SEE ALSO

infer_switching_activity(2)
get_essential_points(2)
report_switching_activity(2)
report_activity(2)

report_exceptions

Generates a report of timing exceptions.

SYNTAX

```
status report_exceptions
  [-from from_list
    | -rise_from rise_from_list
    | -fall_from fall_from_list]
  [-through through_list]*
  [-rise_through rise_through_list]*
  [-fall_through fall_through_list]*
  [-to to_list
    | -rise_to rise_to_list
    | -fall_to fall_to_list]
  [-ignored]
  [-dominant]
  [-nosplit]
```

Data Types

from_list	list
rise_from_list	list
fall_from_list	list
through_list	list
rise_through_list	list
fall_through_list	list
to_list	list
rise_to_list	list
fall_to_list	list

ARGUMENTS

Note: Options marked with asterisks (*) in the SYNTAX can be used multiple times in the same command.

-from *from_list*

Specifies a list of clocks, ports, cells, and pins in the current mode. The report includes only exceptions that have the specified objects in the *from_list*. This option limits the report to information that was set using a path-based command (for example, **set_multicycle_path**) with the **-from** option. The **-from**, **-rise_from**, and **-fall_from** options are mutually exclusive.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking

into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from ***fall_from_list***

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through ***through_list***

Specifies a list of pins or ports. The report includes only paths that go through the pins or ports in *through_list*. This option limits the report to information that was set using a path-based command (for example, **set_multicycle_path**) with the **-through** option. You can use this option multiple times in the same command.

-rise_through ***rise_through_list***

Specifies a list of pins or ports (the same as the **-through** option) except that the paths must have a rising transition at the through points. You can use this option multiple times in the same command.

-fall_through ***fall_through_list***

Specifies a list of pins or ports (the same as the **-through** option) except that the paths must have a falling transition at the through points. You can use this option multiple times in the same command.

-to ***to_list***

Specifies a list of clocks, ports, cells, and pins in the current mode. The report includes only paths that end at the objects in the *to_list* objects. Using this option limits the report to information that was set by using a path-based command (for example, **set_multicycle_path**) with the **-to** option. The **-to**, **-rise_to**, and **-fall_to** options are mutually exclusive.

-rise_to ***rise_to_list***

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to ***fall_to_list***

Same as the **-to** option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-ignored

Lists path timing exceptions that are set on the current mode, but are completely ignored. For example, a false path might be specified from port A to port Z1, but if there is no timing path between those points, the path is ignored. Use **-from** or **-to** to limit the report to certain paths.

-dominant

Lists path timing exceptions that are set on the current mode and are dominant for at least one path. This option can be used with the **-ignored** option to see all exceptions. If neither the **-ignored** or **-dominant** option are given all exceptions are printed whether they affect the paths or not.

-nosplit

Retain long lines, even if they are longer than 80 columns. This is most useful for doing diffs on previous scripts, or for post-processing the script.

DESCRIPTION

This command writes a report showing information about timing exceptions for the current mode. A timing exception is considered dominant if it alters the constraints of a path. If path constraints are different in the absence of a timing exception, the exception alters or dominants the path constraint.

The **report_exceptions** command attempts to identify the reasons an exception is ignored, and classifies these into one of the following: invalid startpoints, invalid endpoints, non-existent paths, or overridden paths. This is reported as separate sections of the report. When **-ignored** is used, the sections show why an exception was ignored.

If a path is unconstrained (that is, if it has a required time of infinity), timing exceptions do not change the path constraints. Applying a timing exception to an unconstrained path does not change the path constraint. A path is unconstrained if it has an infinite required time.

To remove timing exceptions from specified paths, use **reset_path**. **reset_design** removes all attributes from the design, including timing exceptions.

To report only those exceptions that are dominant, use the **-dominant** option. If neither the **-ignored** or **-dominant** option are specified, then all exceptions matching the **-from**, **-through**, or **-to** options are reported and not categorized into ignored and dominant sections.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example lists all timing exceptions set in the current mode.

```
prompt> report_exceptions

****************************************
Report : exceptions
 ...

****************************************


#  e8.tcl, line 9; e8.tcl, line 10
set_multicycle_path 2  -hold -from [get_clocks {clk1}] -to [get_clocks {clk2}]
set_multicycle_path 5  -setup -from [get_clocks {clk1}] -to [get_clocks {clk2}]

#  e8.tcl, line 11
set_false_path -from [get_clocks {clk1}] -to [get_clocks {clk2}]

#  e8.tcl, line 13
set_multicycle_path 2 -through [get_pins {t/Z}]

#  e8.tcl, line 14
set_false_path -through [get_pins {t/A}]

#  e8.tcl, line 16
set_multicycle_path 2 -through [get_pins {u/Z}] -through [get_pins {u/B}]
```

```
#  e8.tcl, line 21
set_false_path -through [get_pins {t/Z}]
```

The following example lists all ignored timing exceptions set in the current mode.

```
prompt> report_exceptions -ignored

****************************************
Report : exceptions
Scenario: default
 ...

****************************************

###############################################################################
## Redundant exceptions (totally overridden by other exceptions):

#  e8.tcl, line 9; e8.tcl, line 10
set_multicycle_path 2  -hold -from [get_clocks {clk1}] -to [get_clocks {clk2}]
set_multicycle_path 5  -setup -from [get_clocks {clk1}] -to [get_clocks {clk2}]

###############################################################################
## Exceptions that do not cover any constrained paths:

#  e8.tcl, line 16
set_multicycle_path 2 -through [get_pins {u/Z}] -through [get_pins {u/B}]

###############################################################################
## Exceptions with no valid -from objects:

#  e8.tcl, line 24
set_false_path -from [get_pins {u/B}]
```

The following example lists all dominant timing exceptions set in the current mode.

```
prompt> report_exceptions -dominant

****************************************
Report : exceptions
Design : counter
Scenario: default
 ...

****************************************

###############################################################################
## Exceptions that are dominant for at least one path:

#  e8.tcl, line 11
set_false_path -from [get_clocks {clk1}] -to [get_clocks {clk2}]

#  e8.tcl, line 13
set_multicycle_path 2 -through [get_pins {t/Z}]

#  e8.tcl, line 14
set_false_path -through [get_pins {t/A}]
```

SEE ALSO

current_design(2)
current_mode(2)
set_false_path(2)
set_max_delay(2)
set_max_time_borrow(2)
set_min_delay(2)
set_multicycle_path(2)

report_extraction_options

Report parasitic extraction options

SYNTAX

string **report_extraction_options**
[-corners]
[-all]

Data Types

list *corners*

ARGUMENTS

-corners

report parasitic extraction options of list of constraint corners

-all

report all parasitic extraction options including default values

DESCRIPTION

Report parasitic extraction options used by rc extraction. By default, it will report user defined extraction options of all constraint corners.

Multicorner-Multimode Support

EXAMPLES

In the following example, the command will print extraction option of current corner

prompt> **report_extraction_options -corners [current_corner]**
1

SEE ALSO

all_corners(2)
current_corner(2)
get_corners(2)
remove_corner(2)
create_corner(2)
create_mode(2)
set_extraction_options(2)
set_parasitic_parameters(2)
report_parasitic_parameters(2)

report_feedthroughs

Creates a report of all feedthrough nets, including the blocks and pins that the nets pass through.

SYNTAX

status **report_feedthroughs**
[-nets *nets*]
[-include_original_feedthroughs]
[-file_name_prefix *prefix_string*]
[-include_buffered]
[-self]
-reporting_style {net_based block_based}

Data Types

nets collection
prefix_string string

ARGUMENTS

-nets *nets*

Specifies the nets for which to report feedthrough pins. For a specified net, the command reports the feedthrough pins that are on the physical net and also within the boundaries of the cell the net belongs to (including the cells in lower hierarchy within the cell). By default, the command reports feedthroughs for all nets.

-include_original_feedthroughs

Reports both the original feedthrough pin and the feedthrough pin inserted by the **place_pins** or **push_down_objects** command. An original feedthrough is any port in a block that is currently on a feedthrough net in the block (where there are 2 or more ports on the net) where that port was not created by any design-planning command (**place_pins** or **push_down_objects**). An original feedthrough port could exist on a pure feedthrough (no logic involved) where 2 or more ports existed because they were included in the Verilog netlist. An original feedthrough port could also exist on a mixed feedthrough net, in which case one of the feedthrough ports on the now feedthrough net was an original port, and the rest were all added by a hierarchical design planning command.

-file_name_prefix *prefix_string*

Prefixes the output file name with *prefix_string*. This option can be used together with the **-reporting_style {block_based}** option to create a common prefix for the individual feedthrough reports. If this option is not specified, the command does not insert a prefix string into the report file name.

-include_buffered

Skips over repeaters and buffers, include buffered feedthroughs when reporting. Without this option, buffered feedthroughs will

be skipped by default.

-self

Reports only current top-level feedthroughs. Without this option, the command reports feedthroughs in all planning-enabled blocks.

-reporting_style ***{net_based block_based}***

Specifies the type of feedthrough report to generate. The **-reporting_style net_based** option generates a single file that contains feedthrough net and block information for all feedthroughs. The **-reporting_style block_based** option generates one file for each block, and each file contains a list of feedthroughs for the block. Default style is **block_based**.

DESCRIPTION

This command reports feedthrough nets and associated block pins for feedthroughs in the design. By default, the command reports all feedthrough nets and pins inserted by the **place_pins** command, but does not include original pins.

By default, the command looks for and reports feedthroughs at multiple levels of physical hierarchy (MPH support).The **set_editability** command can be used to enable/disable reporting on individual reference-blocks or levels of physical hierarchy. For net constraint, the command reports feedthroughs on all nets that originate/begin in an enabled instance/block.

This command supports two types of reports: 1) a net_based report which writes all nets and corresponding block pins to a single file and 2) a block_based report which writes out a separate file for each block with feedthrough pins.

If you specify the **-include_original_feedthroughs** option, the command reports all feedthroughs. To report feedthroughs for only a subset of nets in the design, use the **-nets** option to specify a collection of nets.

Use the **-include_original_feedthroughs** option to include feedthrough nets and pins from the original netlist, in addition to feedthrough nets and pins added by the **place_pins** command. This option is off by default.

The **-file_name_prefix prefix_string** can be used to customize the output file names by prefixing the file name with the specified *prefix_string*. The command inserts a period (.) after the prefix string when creating the file name. This option is off by default. When you specify this option, the command issues a message if the file exists and overwrites the file.

The **-reporting_style** option is a required argument, which consists of 2 choices: *net_based* and *block_based*. **You must choose at least one of these or the command issues a warning message and exits**.

The *net_based* style writes out a single file named *prefix_string*.feedthrough_nets which includes all nets with feedthrough pins. Each feedthrough net record contains a net name and one or more block port names. The feedthrough record is enclosed with curly braces {}.

The *block_based* style writes a single file for each block that contains feedthrough pins. The file name convention is *prefix_string.block_instance_name*.feedthrough_ports, and includes all feedthrough pins that are on the block instance. If the block instance name contains forward slashes (because the path name includes slashes), the forward slashes are replaced with underscore (_) characters to avoid creating unwanted directories.

EXAMPLES

The following example reports feedthrough nets and pins on all nets, and reports using both net_based and block_based styles:

```
prompt> report_feedthroughs -include_original_feedthroughs \
```

-reporting_style {net_based block_based}

The following example reports inserted feedthrough nets and pins on net rstn and reports using the net_based style:

prompt> **report_feedthroughs -reporting_style {net_based} -nets [get_nets rstn]**

SEE ALSO

place_pins(2)
remove_feedthroughs(2)
set_editability(2)

report_first_track_line

Report the name of the track, offset value relative to core area or block boundary and mask constraint of the first trackline for a specified layer and direction of tracks.

SYNTAX

int **report_first_track_line**
-layer *layer*
-dir X | Y
[-relative_to core_area | block_boundary]

Data Types

digits int
layer string

ARGUMENTS

-layer *layer*

Specifies the routing layer to fetch the routing tracks. You can use layer name, layer number, or a collection containing one layer object.

-dir X | Y

Specifies the direction of routing tracks which will be considered for reporting first trackline. The valid values are **X** and **Y**; specify either.

-relative_to core_area | block_boundary

Specifies whether details of first trackline will be reported with respect to core area or block boundary. The valid values are **core_area** and **block_boundary**; specify either. The default is to report first trackline relative to core area.

DESCRIPTION

This command reports name of the track, offset value of first trackline and mask constraint of first trackline relative to core area or block boundary for a specified layer and direction of tracks. For calculation, lower left of core area or block boundary bounding box will be considered.

EXAMPLES

The following example reports detail of first trackline.

```
prompt> report_first_track_line -layer M1 -dir Y
Track name:                  TRACK_1
Track line relative to core_area: 0.0000
Track line mask:              mask_one
1
prompt> report_first_track_line -layer M2 -dir X -relative_to core_area
Track name:                  TRACK_2
Track line relative to core_area: 0.0200
Track line mask:              no_mask
1
prompt> report_first_track_line -layer M1 -dir X -relative_to block_boundary
Track name:                  TRACK_1
Track line relative to block_boundary: 0.0240
Track line mask:              mask_one
1
prompt> report_first_track_line -layer M1 -dir Y
No track line exists as per search criteria
1
```

SEE ALSO

report_tracks(2)
create_track(2)
remove_tracks(2)
get_tracks(2)

report_floorplan_rules

Reports the floorplan rules in the design.

SYNTAX

status **report_floorplan_rules**
[-object_types *type_list*]
[-lib_cells *lib_cells*]
[*rule_list*]

Data Types

type_list list
lib_cells collection
rule_list list

ARGUMENTS

-object_types *type_list*

Specifies the list of object types for which some floorplan rules are defined. Floorplan rules that have specified object types in their *-from_object_types* or *-to_object_types* or *-object_types* will be reported. Valid values for this option are *block_boundary*, *core_area*, *hard_macro*, *placement_blockage*, *routing_blockage*, *shape*, *soft_macro* and *std_cell_area*. This option is mutually exclusive with *rule_list*.

-lib_cells *lib_cells*

Specifies the collection of library cells for which some floorplan rules are defined. Floorplan rules that have specified library cells in their *-from_lib_cells* or *-to_lib_cells* or *-lib_cells* options are reported. This option is mutually exclusive with the *rule_list* option.

rule_list

Specifies the list of floorplan rule names to report. This option is mutually exclusive with the *-object_types* and *-lib_cells* options.

DESCRIPTION

The **report_floorplan_rules** command generates a report of existing floorplan rules in a design based on specified input parameters. For a matching floorplan rule, the command reports all applicable parameters of that rule in a tabular format. It will not report those parameters that are not applicable for that rule. On successful reporting the command returns a status of 1.

The *rule_list* option is mutually exclusive with other options *-object_types* and *-lib_cells*. If a named rule in *rule_list* does not exist or

no rule exists for the specified object types or library cells, the tool issues a warning message.

EXAMPLES

The following example tries to report two floorplan rules named e2 and ab out of which ab does not exist.

```
prompt> report_floorplan_rules {e2 ab}

Warning: Floorplan rule 'ab' does not exist.
****************************************
Report : report_floorplan_rules
Design : ABCD
Version: M-2016.12-SP2-BETA
Date   : Thu Jan 12 12:29:56 2017
****************************************

-------------------------------------------------------------------
Name            Type     Parameters        Value
-------------------------------------------------------------------
e2              enclosure Max Value        214748
                  Min Value         0
                  Offset            0
                  Step              0
                  Valid Range       3 - 20
                  Forbidden List    4, 6, 8
                  From Object Types  std_cell_area
                  To Lib Cells      lc1, lc2, lc3
                  Must Enclose      true
                  Sides             vertical

1
```

The following example tries to report floorplan rules for the std_cell_area object type and library cell lc4. Since no floorplan rule exists for library cell lc4, the tool issues a warning message and prints floorplan rule for object type std_cell_area.

```
prompt> report_floorplan_rules -object_types std_cell_area -lib_cells \
  [get_lib_cells */lc4]

Warning: No floorplan rules exists for given lib cells.
****************************************
Report : report_floorplan_rules
Design : ABCD
Version: M-2016.12-SP2-BETA
Date   : Thu Jan 12 12:29:56 2017
****************************************

-------------------------------------------------------------------
Name            Type     Parameters        Value
-------------------------------------------------------------------
e2              enclosure Max Value        214748
                  Min Value         0
                  Offset            0
                  Step              0
```

```
                    Valid Range        3 - 20
                    Forbidden List     4, 6, 8
                    From Object Types  std_cell_area
                    To Lib Cells       lc1, lc2, lc3
                    Must Enclose       true
                    Sides              vertical

    1
```

SEE ALSO

remove_floorplan_rules(2)
set_floorplan_area_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_extension_rules(2)
set_floorplan_forbidden_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_length_rules(2)
set_floorplan_spacing_rules(2)
set_floorplan_width_rules(2)

report_frame_properties

Reports the frame view properties previously extracted and annotated by the **create_frame** command.

SYNTAX

status **report_frame_properties**
-library *library_name*
-output *file_name*
[-block *block_name*]
[-implant_width true | false]
[-diffusion_width_height true | false]
[-source_drain_annotation true | false]

Data Types

lib_name string
file_name string
block_name string

ARGUMENTS

-library *library_name*

Specifies name of the library containing the frame views.

-output *file_name*

Writes the report to the specified file.

-block *block_name*

Specifies the name of the frame block from which to report the properties, with the ".frame" extension. If this option is omitted, report the properties from all frame blocks in the specified library.

-implant_width *true* | *false*

Reports the implant width property. The value is true by default.

-diffusion_width_height *true* | *false*

Reports the diffusion width/height property. The value is true by default.

-source_drain_annotation *true* | *false*

Reports source-drain annotation property. The value is true by default.

If you omit all three property type options from the command, it reports all three properties.

DESCRIPTION

This command reports the properties previously extracted by the **create_frame** command and annotated on the frame views in a specified library. You must specify the library to be reported. To report a specific frame view in the library, use -block for the block name. If no -block option specified, reports all blocks frame view in the library.

You can set any one or more of the options **-implant_width**, **-diffusion_width_height**, and **-source_drain_annotation** to **false** to turn off their respective reports, or omit any of the three options from the command to report the property respectively.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command reports the properties of the "OAI" block in the "io" library:

```
prompt> report_frame_properties -library io -block OAI.frame

****************************************
Report : frame properties
Library: io
 ...
*************************
Design : OAI
*************************
 Total number of row: 1.

***** implant_width property *****
    ---------------------------------------------------
    Position              Layer          Width
    ---------------------------------------------------
    Row(1) Lower-left     VTL_N(10)      1.6000
    Row(1) Upper-left     VTL_P(11)      1.6000
    Row(1) Lower-right    VTL_N(10)      1.6000
    Row(1) Upper-right    VTL_P(11)      1.6000

***** diffusion_width_height property *****
    ---------------------------------------------------
    Position              Width          Height
    ---------------------------------------------------
    Row(1) Lower-left     0.2000         0.1100
    Row(1) Upper-left     0.2000         0.1100
    Row(1) Lower-right    0.2000         0.1100
    Row(1) Upper-right    0.2000         0.1100

***** source_drain annotation ******
    ----------------------------------------
    Position                annotation
    ----------------------------------------
```

```
    Row(1) Lower-left       Source
    Row(1) Upper-left       Source
    Row(1) Lower-right      Drain
    Row(1) Upper-right      Drain
```

If no properties were previously annotated on the frame view by the **create_frame** command, the report looks like this:

```
***** implant_width property *****
    No implant width property on this block.

***** diffusion_width_height property *****
    No diffusion width height property on this block.

***** source_drain annotation ******
    No source drain annotation on this block.
```

SEE ALSO

create_frame(2)

report_freeze_ports

Reports on the **freeze_clock_ports** and **freeze_data_ports** attributes on cells which prevent synthesis from modifying their port signature during optimization.

SYNTAX

string **report_freeze_ports** [-all] [*cell_list*]

list *cell_list*

ARGUMENTS

-all

When given, all cells with either clock or data port-freeze specifications is listed.

cell_list

Specifies a list of cells on which to report the **freeze_clock_ports** and **freeze_data_ports** attributes.

DESCRIPTION

Reports on the **freeze_clock_ports** and **freeze_data_ports** attributes on cells in the current design which prevent these cells from having ports of the given class added or removed during optimization.

EXAMPLES

The following command reports on the freeze_ports settings of the "block1" and "analog1" cells for use during synthesis.

```
prompt> report_freeze_ports -data [get_cells {block1 analog1}]
Corner default:
  0 process number, 0 process label, 0 voltage, and 0 temperature mismatches.
  0 cells affected for early, 0 for late.
  0 voltage and 0 temperature extrapolations.

Report : freeze_ports

Instance u_block1 (block1)
```

```
 data clock

Instance u_analog1 (analog1)
 data clock
2 of 24 instances are freeze_ports
1

1
```

SEE ALSO

get_attribute(2)

report_fsm

Report failsafe FSM related information. By default it would be: Header Design:FSM, Encoding, State Vector(bus name), Attribute

SYNTAX

string **report_fsm**
[-all]
[-design *module*]
[-verbose]
[-show_error_states]
[-state_transitions]

module *string*

ARGUMENTS

-all

If -all is provided, print all available FSM modules in design. It will overwrite -design option.

-design *string*

If module name is provided with -design, report information on the module. Either -all or -design needs to be provided.

-verbose

Print additional following messages. Design: Name of the module the FSM is identified in; FSM Name: Name of the FSM; Filename: Corresponding filename and FSM line number; Clock: Name of the signal connected to the SEQGEN clock pins; Async Reset: Name of the signal(s) connected to the SEQGEN async reset pins. Put "unspecified" if there isn't any; Encoding: Number of bits in the identified state register and show current encoding style; State Vector: List of register names in order of the state vector including state bits and parity bits; Synchronizer: Name for synchronizer used in Hamming2; FSM_COMPLETE: True / False (could the RTL or TCL setting); Re-Encoding: If re-encoding is set either from RTL or TCL, display the target(h2/h3); otherwise, None; State Encodings: show enumerated state names and state encodings (show parity bits also if FSM is encoded); Default Next State: Show the name of the default next state per the names in the State Encodings section; Default States: Print states that is not in state vector if FSM is not fully enumerated; Tool Inferred Error: Show all other possible states that are illegal under the current encoding;

-show_error_states

Optional flag to show illegal encoding and behavior. In case of hamming2/hamming3, show state name, current encoding, original encoding and all possible 1-bit error for each enumerated state.

-state_transitions

Print all possible state vector transitions.

attributes

fc - FSM_COMPLETE
h2 - Hamming2 reencoding
h3 - Hamming3 reencoding
notfusa - Not fully enumerated & no fsm_complete

DESCRIPTION

The command **report_fsm** can be run throughout the Synopsys Automotive Flow and reports failsafe FSM related information.

EXAMPLES

The following are examples of the textual report generated by **report_fsm**:

```
prompt> report_fsm
FSM_RTL:state   hamming2    state_reg[2:0]    fc, h2

prompt> report_fsm -verbose
Design    : FSM_RTL
FSM Name  : state
Filename  :
Clock   : clk
Async Set : unspecified
Async Reset : r
Current Encoding(length / style): 3 / hamming2
State Vector  : { state_reg[0] state_reg[1] state_reg[2] }
Synchronizer Register :
FSM_COMPLETE    : True
Re-Encoding   : hamming2
State Encodings and Order
fsm_state_0   : 1001
fsm_state_1   : 0000
fsm_state_2   : 1111
fsm_state_3   : 1010
fsm_state_4   : 0101
fsm_state_5   : 0011
Default Next State  : fsm_state_5

Tool Inferred States (Mapping: State Encoding)
Default States  : 0110, 1100
Tool Inferred Error : 0001, 0010, 0100, 0111, 1000, 1011, 1101, 1110

prompt> report_fsm -show_error_states
FSM_RTL:state   hamming2    state_reg[2:0]    fc, h2
State Encodings and Order (Original Encoding) (Error Encoding):
fsm_state_0 : 1001  (100)   (0001, 1101, 1011, 1000)
fsm_state_1 : 0000  (000)   (1000, 0100, 0010, 0001)
fsm_state_2 : 1111  (111)   (0111, 1011, 1101, 1110)
fsm_state_3 : 1010  (101)   (0010, 1110, 1000, 1011)
```

```
fsm_state_4 : 0101  (010)   (1101, 0001, 0111, 0100)
fsm_state_5 : 0011  (001)   (1011, 0111, 0001, 0010)
```

SEE ALSO

report_global_timing

Reports a top-level summary of the design timing.

SYNTAX

status **report_global_timing**
[-delay_type max | min]
[-groups *group_list*]
[-include *include_list*]
[-format *format_spec*]
[-output *file_name*]
[-separate_non_standard_groups]
[-separate_all_groups]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-significant_digits *digits*]
[-pba_mode none | path | exhaustive]

Data Types

group_list	list
include_list	list
format_spec	string
file_name	string
mode_list	list
corner_list	list
scenario_list	list
digits	integer

ARGUMENTS

-delay_type max | min

Specifies the type of timing delays for which violations are to be reported. By default, both setup (max) or hold (min) violations are reported.

-groups *group_list*

Reports only the specified path groups.

-include *include_list*

Specifies a list of additional reporting options. *include_list* is a list that includes one or more of the following reporting options.

- **inter_clock** - Shows the breakdown of violations for combinations of launching and capturing clocks.
- **non_violated** - Shows clock combinations that have no violations. By default, only the worst violating timing path to each endpoint is considered in the report.
- **per_clock_violations** - Takes into account other paths to the same endpoint differentiated by launching, capturing clocks and path groups.
- **scenario_details** - In multi-scenario analysis, shows the breakdown of violations shown in the merged report by scenario.

-format *format_spec*

Specifies the reporting format. *format_spec* follows the format {[narrow | wide] [csv]} The following values are allowed:

- **narrow** (default) - Generates a narrow report.
- **wide** - Generates a wide report.
- **csv** - Generates a CSV report. When using this option, specify the output file name with the **-output** option.

-output *file_name*

Specifies the output file name for CSV report.

-separate_non_standard_groups

Divides the report into subreports separating data for standard path groups and for the following nonstandard path groups: **default**, **async_default**, **clock_gating_default**.

-separate_all_groups

Divides the report into subreports showing data for all path groups which have violations separately.

-modes *mode_list*

Specifies the modes to be used when generating the report. Data from each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option will be included in the report. If this option is not given, the report will include data from scenarios of all modes. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the report will include data from every active scenario of the design. It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners to be used when generating the report. Data from each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option will be included in the report. If this option is not given, the report will include data from each scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios to be used when generating the report. Data from each of the specified scenarios is included in the report. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the report will include data from every scenario of the design.

-significant_digits digits

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13. If you do not use this option, the number of digits is specified by the **shell.common.report_default_significant_digits** application option, which defaults to 2. Use this option if you want to override the default.

-pba_mode none | path | exhaustive

Controls path-based analysis with the following modes:

- **none** (the default) - Path-based analysis is not applied.
- **path** - Path-based analysis is applied over the paths that gave the worst graph-based analysis violation. There is no guarantee that the reported path-based analysis slack over every endpoint is the worst one.
- **exhaustive** - An exhaustive path-based analysis path search algorithm is applied to determine the worst path-based analysis slack violations.

DESCRIPTION

The **report_global_timing** command generates a top-level summary of the timing for the design. The report shows the worst negative slack per endpoint, the sum of all worst negative slacks per endpoint, and a number of violating endpoints. By default, each violating endpoint is only counted one time and only one worst slack at each violating endpoint contributes to TNS and WNS. With the **-groups** option, each endpoint contributes to the worst negative slack and total negative slack of each group in which it has a violating slack.

The summary information is broken down into several columns, showing violations for register-to-register, input-to-register, input-to-output, and register-to-output timing paths. WNS doesn't report the worst slack for each category, it reports the worst slack per endpoint and categorize those worst slacks by reg2reg, in2reg, reg2out and in2out. This behavior aligns with PT report.

Multicorner-Multimode Support

EXAMPLES

The following example generates a report of violations in the current design for path groups whose name starts with 'clk'.

```
prompt> report_global_timing -groups [get_path_group CLK*]
****************************************
Report : global_timing
        -groups CLK1 CLK0
 ...
****************************************

Setup violations
-------------------------------------------------------------
        Total  reg->reg   in->reg  reg->out   in->out
-------------------------------------------------------------
WNS     -20.34    -20.34      0.00     -1.19     -0.98
TNS   -6887.98  -6882.84      0.00     -3.17     -1.96
NUM       398       392         0         4         2
-------------------------------------------------------------

Hold violations
-------------------------------------------------------------
        Total  reg->reg   in->reg  reg->out   in->out
-------------------------------------------------------------
WNS      -3.55     -3.55      0.00      0.00      0.00
TNS    -561.17   -561.17      0.00      0.00      0.00
NUM       418       418         0         0         0
-------------------------------------------------------------
```

The second example shows a (truncated) report subdivided into subreports for all launching clock and capturing clock combinations. Each subreport shows a C2C line identifying the clock combination. The launching clock name is shown first, followed by '->' and the name of the capturing clock. The symbol "*" means any clock.

```
prompt> report_global_timing -include {inter_clock}
****************************************
Report : global_timing
        -include {inter_clock}
 ...
****************************************

Setup violations
------------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
------------------------------------------------------------
C2C * -> *
WNS     -26.57    -26.57      0.00     -1.72     -0.98
TNS  -33377.09 -33369.93      0.00     -5.19     -1.96
NUM       1958      1950         0         6         2
------------------------------------------------------------
C2C * -> CLK0
WNS     -20.34    -20.34      0.00     -1.19     -0.98
TNS   -6889.55  -6884.41      0.00     -3.17     -1.96
NUM        403       397         0         4         2
------------------------------------------------------------
C2C CLK0 -> CLK0
WNS     -20.34    -20.34      0.00      0.00      0.00
TNS   -6884.25  -6884.25      0.00      0.00      0.00
NUM        396       396         0         0         0
------------------------------------------------------------
C2C AUDIO_SCLK_IN -> CLK0
WNS      -1.10      0.00      0.00     -1.10     -0.98
TNS      -3.06      0.00      0.00     -1.10     -1.96
NUM          3         0         0         1         2
------------------------------------------------------------
 ...
```

In the third example, **-include {per_clock_violations}** is added. Comparing the report output from the second example, more violations are reported because more violating paths to the same endpoint are taking into account.

```
prompt> report_global_timing -include {inter_cl per_cl}
****************************************
Report : global_timing
        -include { inter_clock per_clock_violations }
 ...
****************************************

Setup violations
------------------------------------------------------------------
          Total   reg->reg   in->reg   reg->out   in->out
------------------------------------------------------------------
C2C * -> *
WNS      -26.57     -26.57      -2.17      -1.72      -0.98
TNS  -391171.34 -391157.25      -2.17      -9.81      -2.05
NUM       57766      57751          1         11          3
------------------------------------------------------------------
C2C * -> CLK0
```

```
WNS     -20.34    -20.34      0.00     -1.72     -0.98
TNS   -39827.79  -39817.95     0.00     -7.79     -2.05
NUM       5256      5244        0        9         3
-----------------------------------------------------------------
C2C CLK0 -> CLK0
WNS     -20.34    -20.34      0.00      0.00      0.00
TNS    -6884.25   -6884.25     0.00      0.00      0.00
NUM        396       396        0         0         0
-----------------------------------------------------------------
C2C ADC_PCMCLK_IN -> CLK0
WNS      -0.09      0.00      0.00      0.00     -0.09
TNS      -0.09      0.00      0.00      0.00     -0.09
NUM          1         0         0         0         1
-----------------------------------------------------------------
C2C AUDIO_SCLK_IN -> CLK0
WNS      -1.10      0.00      0.00     -1.10     -0.98
TNS      -3.06      0.00      0.00     -1.10     -1.96
NUM          3         0         0         1         2
-----------------------------------------------------------------
C2C ADC_SCLK_IN -> CLK0
WNS      -1.19      0.00      0.00     -1.19      0.00
TNS      -1.19      0.00      0.00     -1.19      0.00
NUM          1         0         0         1         0
-----------------------------------------------------------------
C2C C_D950/C_PCU/C_DPCUB_DP/C_RIDP/I15/I_0_0_15/Q -> CLK0
WNS      -7.26     -7.26      0.00      0.00      0.00
TNS    -1775.25   -1775.25     0.00      0.00      0.00
NUM        350       350        0         0         0
-----------------------------------------------------------------
 ...
```

The following example shows the use of alternative reporting format styles. The report is simultaneously printed out in a wide format and written into a CSV file:

```
prompt> report_global_timing  -format {csv wide} -output ./example.csv
****************************************
Report : global_timing
       -format { wide csv }
       -output example.csv
 ...
****************************************


Setup violations
        Total      |        reg->reg      |    in->reg  |      reg->out  |     in->out    |
 WNS      TNS    NUM |  WNS      TNS   NUM  | WNS  TNS  NUM | WNS    TNS  NUM | WNS    TNS  NUM |
-------------------------------------------------------------------------------------------------
-26.57 -33377.09 1958 | -26.57 -33369.93 1950 | 0.00 0.00   0 | -1.72 -5.19  6  | -0.98 -1.96   2 |

Hold violations
        Total      |        reg->reg      |    in->reg  |      reg->out  |     in->out    |
 WNS      TNS    NUM |  WNS      TNS   NUM  | WNS  TNS  NUM | WNS   TNS  NUM  | WNS    TNS  NUM |
-------------------------------------------------------------------------------------------------
-10.89   -757.51  532 | -10.89   -757.51  532 | 0.00 0.00   0 | 0.00  0.00  0   |  0.00  0.00   0 |

prompt> sh cat ./example.csv
Type,Total_WNS,Total_TNS,Total_NUM,reg2reg_WNS,reg2reg_TNS,reg2reg_NUM,in2reg_WNS,in2reg_TNS,
in2reg_NUM,reg2out_WNS,reg2out_TNS,reg2out_NUM,in2out_WNS,in2out_TNS,in2out_NUM
```

```
max,-26.57,-33377.09,1958,-26.57,-33369.93,1950,0.00,0.00,0,-1.72,-5.19,6,-0.98,-1.96,2
min,-10.89,-757.51,532,-10.89,-757.51,532,0.00,0.00,0,0.00,0.00,0,0.00,0.00,0
```

The following example shows the report with separate results for each path group:

```
prompt> report_global_timing -groups {CLK0 AUDIO_SCLK_IN} \
        -separate_all_groups -delay_type max
 ...

Setup violations for paths in CLK0
-------------------------------------------------------------
       Total  reg->reg   in->reg  reg->out   in->out
-------------------------------------------------------------
WNS    -20.34   -20.34     0.00    -1.19    -0.98
TNS  -6887.98 -6882.84     0.00    -3.17    -1.96
NUM       398      392        0        4        2
-------------------------------------------------------------

Setup violations for paths in AUDIO_SCLK_IN
-------------------------------------------------------------
       Total  reg->reg   in->reg  reg->out   in->out
-------------------------------------------------------------
WNS     -2.31    -2.31     0.00     0.00     0.00
TNS    -47.60   -47.60     0.00     0.00     0.00
NUM        32       32        0        0        0
-------------------------------------------------------------
```

The following example shows the report with separate results for standard and nonstandard path groups:

```
prompt> report_global_timing -groups {CLK0 AUDIO_SCLK_IN *async*} \
        -separate_non_standard_groups  -delay_type max
 ...

Setup violations for standard paths
-------------------------------------------------------------
       Total  reg->reg   in->reg  reg->out   in->out
-------------------------------------------------------------
WNS    -20.34   -20.34     0.00    -1.19    -0.98
TNS  -6935.58 -6930.45     0.00    -3.17    -1.96
NUM       430      424        0        4        2
-------------------------------------------------------------

Setup violations for paths in **async_default**
-------------------------------------------------------------
       Total  reg->reg   in->reg  reg->out   in->out
-------------------------------------------------------------
WNS    -23.71   -23.71     0.00     0.00     0.00
TNS  -1010.10 -1010.10     0.00     0.00     0.00
NUM       186      186        0        0        0
-------------------------------------------------------------
```

If you specify the **-pba_mode exhaustive** option, path-based analysis is used to calculate the violating slack values. This option requires that a path search be performed to ensure that the returned slack values are truly the worst. The additional runtime required for this option depends on the amount of timing improvement resulting from path-based analysis. As the timing improvement from path-based analysis increases, the runtime for the path search increases.

SEE ALSO

report_timing(2)

report_grids

Reports step, offset, allowed orientations, and other information for the currently defined grids in the design.

SYNTAX

status **report_grids**
[-ref_blocks *design_list*]
[-type block | user | finfet]
[*grid*]

Data Types

design_list collection
grid collection

ARGUMENTS

-ref_blocks *design_list*

Specifies the block reference designs to report block grid information. This option is applicable only to block grids.

-type block | **user** | **finfet**

Specifies the type of grid to report. Specify *block*, for block grids or *user*, for user grids or *finfet* for finfet grid defined in the technology section.

grid

Specify the name of the grid to report. You can specify a single object or a collection of grid names. By default, the command reports all grids.

DESCRIPTION

This command reports the grid information for the specified grid or the specified block design references. By default, all grids in the current design are reported. For block grids, their association with block design references are reported and if any block cell instance is off the associated block grid, an error would be reported.

EXAMPLES

The following example reports all grids in the current design:

```
prompt> report_grids
```

The following example reports all block grids in the current design:

```
prompt> report_grids -type block
```

The following example reports all user grids in the current design:

```
prompt> report_grids -type user
```

The following example uses the *grid* argument to report all grids with name matching gr:

```
prompt> report_grids [get_grids gr*]
```

SEE ALSO

create_grid(2)
get_grids(2)
remove_grids(2)
set_block_grid_references(2)
set_grid(2)
snap_cells_to_block_grid(2)

report_groups

Reports groups in the current design.

SYNTAX

int **report_groups**
[-verbose]
[-nosplit]
[-hdl_of_module *module*]
[*group_list*]

Data Types

module	string
group_list	list

ARGUMENTS

-verbose

Displays verbose group information.

-nosplit

Retains long lines and does not split lines if column overflows.

-hdl_of_module

Reports the HDL groups contained in the specified module.

group_list

Specifies a list of groups to report. The list can contain group names, patterns, or collections. A collection can be specified by using the **get_groups** command. Support hierarchical pattern when **-hdl_of_module** option is specified.

DESCRIPTION

The **report_groups** command generates a report about the groups in the design. The report includes the group name and a description of the group. If **-verbose** is specified, the report also includes additional attributes and a list of objects contained in the group.

If *group_list* is specified, those groups are reported. If it is not specified, then all groups in the design are reported.

EXAMPLES

The following example reports all groups in the design.

```
prompt> report_groups
Group           Description
--------------------------------------------------------------------------------
GROUP_0         Contains 3 objects   type: set
MY_GROUP          Contains 2 objects   type: set
1
```

The following example reports HDL group named blk2 inside blk1 in the module ORCA.

```
prompt> report_groups -hdl_of_module ORCA blk1/blk2
Group           Description
--------------------------------------------------------------------------------
blk1/blk2        Contains 4 objects   type: set
1
```

The following example reports verbose information for the group named "GROUP_0".

```
prompt> report_groups -verbose GROUP_0
Group           Description
--------------------------------------------------------------------------------
GROUP_0         Contains 3 objects   type: set
                remove_when:          no_auto_removal
                allow_duplicate_name:   false
                objects:              n13
                                      n12
                                      POLYGON_14_1
1
```

The following example reports verbose information HDL group named blk2 inside blk1 in the module ORCA.

```
prompt> report_groups -verbose -hdl_of_module ORCA blk1/blk2
Group           Description
--------------------------------------------------------------------------------
blk1/blk2        Contains 4 objects   type: set
                remove_when:          no_auto_removal
                allow_duplicate_name:   false
                cells:                reg13
                                      gen12
                                      myreg


1
```

SEE ALSO

add_to_group(2)
create_group(2)
get_groups(2)

remove_groups(2)
remove_from_group(2)

report_gui_stroke_bindings

Print a report on the dictionaries and the stroke-command bindings they contain..

SYNTAX

report_gui_stroke_bindings
[-dictionary *dictionary_name*]

ARGUMENTS

-dictionary *dictionary_name*

Specify which dictionary should have its bindings reported. If not specified then all dictionaries will be reported.

DESCRIPTION

This command takes the data returned by get_gui_stroke_bindings and formats it into a report that is human-readable. The report prints out the stroke-to-command bindings for all dictionaries.

This command is defined as a part of the strokes Tcl package, and the command is automatically imported into the global namespace when the package is loaded.

EXAMPLES

```
ca_shell> report_gui_stroke_bindings

Dictionary: Graphics

stroke      cmdType      cmd
------      -------      ---
5           builtin      Pan_Center_Rect
852         builtin      Zoom_Full
258         builtin      Zoom_Full
159         builtin      Zoom_Out_Rect
357         builtin      Zoom_Out_Rect
753         builtin      Zoom_In_Rect
951         builtin      Zoom_In_Rect
```

```
654         builtin     Pan_Center_Rect
456         builtin     Pan_Center_Rect
123698741   builtin     Zoom_In_Rect
147896321   builtin     Zoom_In_Rect
s:14789     tcl_cmd     myTclCmd
14789       tcl_cmd     myTclCmd %rect
```

SEE ALSO

set_gui_stroke_preferences(2)
set_gui_stroke_binding(2)
report_gui_stroke_builtins(2)

report_gui_stroke_builtins

Print a report on the non-Tcl commands available for stroke bindings..

SYNTAX

report_gui_stroke_builtinss
[-dictionary *dictionary_name*]

ARGUMENTS

-dictionary *dictionary_name*

Specify which dictionary should have its builtins reported. If not specified then only the global builtins will be printed.

DESCRIPTION

This command takes the data returned by get_gui_stroke_bindings and formats it into a report that is human-readable. The report prints out the non-Tcl commands that can be used in stroke bindings.

This command is defined as a part of the strokes Tcl package, and the command is automatically imported into the global namespace when the package is loaded.

EXAMPLES

ca_shell> report_gui_stroke_builtins

Built-In Commands:

name

Zoom_In_Rect
Zoom_Out
Pan_Center_Rect
Zoom_Out_Rect
Zoom_Full
Zoom_In

SEE ALSO

set_gui_stroke_preferences(2)
set_gui_stroke_binding(2)
report_gui_stroke_bindings(2)

report_hdl_libraries

Lists HDL (RTL) libraries and their directory locations.

SYNTAX

status **report_hdl_libraries**
[-nosplit]
[*library_list*]

Data Types

library_list list

ARGUMENTS

library_list

Specifies a list of HDL library names whose mappings are to be displayed. The default is to display all HDL libraries.

DESCRIPTION

The **report_hdl_libraries** command lists the directory locations of specified libraries.

An HDL library is where the tool places the intermediate files created by the **analyze** RTL analysis command.

EXAMPLES

The following example reports the location of the "WORK" HDL library.

```
prompt> report_hdl_libraries
****************************************
Report : hdl libraries
Version: P-2019.03-SP4
Date   : Mon Aug 26 06:59:52 2019
****************************************
WORK
   (/home/user/HDL_LIBRARIES/WORK)
```

1

SEE ALSO

analyze(2)
define_hdl_library(2)
elaborate(2)

report_hierarchy

Displays logical and physical reference hierarchy for a block.

SYNTAX

string **report_hierarchy**
[-block block_name]
[-physical_context]
[-nosplit]
[-hierarchical]
[-no_leaf]

Data Types

block_name string formatted [libName:]designName[/labelName][.viewName]

ARGUMENTS

-block

Specifies a block to generate the report on. If argument is not provided, the report is generated for the current block.

-physical_context

Indicates report should only be generated on physical blocks and leaf modules. By default, the report also displays internal module references.

Because internal modules are not shown, references of internal modules are displayed as references of their enclosing block.

-nosplit

Prevents line splitting when a field overflows.

-hierarchical

Displays the unfolded hierarchy. By default, multiply instantiated blocks or modules are reported only once.

-no_leaf

Indicates leaf modules are to be excluded from the report.

DESCRIPTION

The **report_hierarchy** command generates a report which displays all reference blocks and internal modules for a block. The command's default behavior is to generate a report for the current block displaying both the physical and logical reference hierarchy to the leaf module level. Multiple occurrences of a block or module are folded unless they are bound to different views. To differentiate blocks from internal modules, the full block name ([libName:]designName[/labelName][.viewName]) is printed. Leaf modules are displayed with their corresponding library name to their right.

EXAMPLES

```
prompt> report_hierarchy
****************************************
Report : hierarchy
 ...

****************************************

tiny:TOP.design
 tA
   AN2LVTD0                    tcbn90glvt
   tB
     tiny:MID.design
       mA
         mB
           tiny:BOT.design
             A
               B
                 C
                   D
             E
               OR2LVTD0        tcbn90glvt
             F
           mC
             mD
       mE
       mF
     tC
       INVLVTD0                tcbn90glvt
       tD
         XOR2LVTD0             tcbn90glvt
 tE
 tF

prompt> report_hierarchy -physical_context -no_leaf
****************************************
Report : hierarchy
 ...

****************************************

tiny:TOP.design
 tiny:MID.design
   tiny:BOT.design
```

SEE ALSO

report_references(2)

report_host_options

Reports settings for multi-threaded and distributed processing created by **set_host_options**.

SYNTAX

status **report_host_options**
[-nosplit]

ARGUMENTS

-nosplit

Retains long lines in the output, even if the line is longer than 80 columns. By default, the command inserts line continuation characters (\) to break long lines.

DESCRIPTION

This command reports the distributed computing options set by the **set_host_options** command.

EXAMPLES

The following example creates a host option named block_script, then reports the settings for the host option.

```
prompt> set_host_options -name block_script localhost
1

prompt> report_host_options
****************************************
Report : host_options
 ...

****************************************

Max_cores: 1

Global Host Options:
Attribute        Value
```

```
----------------------------------------
max_cores         1
num_processes     1
submit_command    rsh
work_dir          ./work_dir
target            ALL

Host Options 'block_script':
Attribute         Value
----------------------------------------
name              block_script
submit_command    rsh
work_dir          ./work_dir
target            ALL
host_names        localhost
1
```

SEE ALSO

set_host_options(2)
remove_host_options(2)

report_hot_spots

Reports voltage drop related cell attributes in the columnized format for a hot spot grid box, so it is easy for users to identify root causes. This command has to be invoked after the **generate_hot_spots** command.

SYNTAX

status **report_hot_spots**
[-index *index*]
[-summary]
[-object *cell_instance_object*]
[-verbose]

Data Types

net_name integer
object collection

ARGUMENTS

-index *index*

Specifies the grid hot spot index for which voltage drop related attributes are reported. This option is mutually exclusive with the -summary or -object option.

-summary

Displays a summary of the report for grid boxes, such as the number of grids, the number of instances, and the number of grids over the percentage threshold. This option is mutually exclusive with -index or -object option.

-object *cell_instance_object*

Reports on the hot spot grid which contains the specified cell instance object. This option is mutually exclusive with the -index or -summary option.

The **report_hot_spots** command reports voltage drop and relevant attributes which are useful to identify voltage drop root causes, such as current, overlap current, effective resistance, timing window, slack, output load and static power. Those attributes are reported in columnized format, so it is easy to allow users to identify voltage drop root causes. This command has to be invoked after the **generate_hot_spots** command. For an instance in a grid box, the overlap current is estimated by accumulating timing-window-based current cross its neighboring cells.

EXAMPLES

The following is an example of a verbose report:

```
prompt> generate_hot_spots -net VDD
prompt> report_hot_spots -index 3 -verbose
Voltage drop hot spot index: 3
Grid BBox: [2.57e+03 1.23e+03][2.58e+03 1.23e+03]
Number of instances: 4
Max effective voltage drop: 0.076524 Grid Rank 4/328323

Neighboring grids by index
--------------------------------------------------------------------------------
  2628    1032    124    287    5120   14651
 14466   13717    529   1551   11162   18418
 12900    1106     73     80     305    8549
  2832     369      4      3      19     195
698569      44      8     14     228    5020
 17526    1968    414    447    7004   12169

--------------------------------------------------------------------------------

Top instances sorted by instance effective voltage drop (exclude 0 physical only cells)
--------------------------------------------------------------------------------
Instance                        EVD     Current  Overlap_I Eff R   Timing  window  Slack   Slew    Load    Power
lsw2/lsw2_regsfile/latch902      0.085   0.524    3.37     0.00545  [0.999   1.26] 0.0814  0.048   0.164   0.0971
lsw2/lsw2_regsfile/U62331        0.0818  2.89e-05 0.0012   0.00623  [2.46    4.31] -0.00221 0.0163  0.0404  3.41e-05
lsw2/lsw2_regsfile/U153273       0.0798  2.67e-05 1.2      0.00624  [1.21    2.31] 0.122   0.0326  0.0636  2.44e-05
lsw2/lsw2_regsfile/U153272       0.0795  6.89e-05 0.587    0.00595  [1.26    2.42] 0.115   0.00732 0.012   6.52e-05
--------------------------------------------------------------------------------
Total accumulated current in timing window (1.21, 1.26): 0.524206
Total accumulated current: 0.524303

Accumulated current in timing window:
--------------------------------------------------------------------------------
[    0.999      1.21 ) : 0.524
[     1.21      1.26 ) : 0.524
[     1.26      2.31 ) : 9.56e-05
[     2.31      2.42 ) : 6.89e-05
[     2.42      2.46 ) : 2.9e-08
[     2.46      4.31 ) : 2.89e-05
--------------------------------------------------------------------------------
Cells with overlap timing window (accumulated current: 0.524)
--------------------------------------------------------------------------------
lsw2/lsw2_regsfile/latch902      0.524
lsw2/lsw2_regsfile/U153272       6.89e-05
lsw2/lsw2_regsfile/U153273       2.67e-05
--------------------------------------------------------------------------------
```

SEE ALSO

report_hot_spots(2)
remove_hot_spots(2)

report_ideal_network

Displays information about ports, pins, nets, and cells on ideal networks in the current design.

SYNTAX

status **report_ideal_network**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-net]
[-cell]
[-load_pin]
[-timing]
[*object_list*]

Data Types

object_list list

ARGUMENTS

-modes *mode_list*

Specifies the modes to be reported. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is reported separately. If this option is not given, the command reports scenarios of the current mode. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners to be reported. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is reported separately. If this option is not given, the command reports scenarios of the current corner. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios to be reported. Each of the given active scenarios is reported. It is an error to give this option with the **-modes** or **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any).

-net

Displays all nets in the specified ideal networks. By default, nets are displayed for all ideal networks.

-cell

Displays all cells in the specified ideal networks. By default, cells are displayed for all ideal networks.

-load_pin

Specifies to display load pins (boundary pins) of the specified ideal networks along with any ideal timing set on them using the **set_ideal_latency** and **set_ideal_transition** commands. By default, load pins are displayed for all ideal networks.

-timing

Specifies to display all internal pins (non-source and non-boundary pins) in the specified ideal networks that have ideal timing set on them using the **set_ideal_latency** and **set_ideal_transition** commands. By default, internal pins with timing are displayed for all ideal networks.

object_list

Defines a list of source ports or source pins of the specified ideal networks to be displayed. By default, the source pins and source ports for all ideal networks in the current design are displayed.

DESCRIPTION

Displays information about the ideal networks in the current design. If no arguments are specified, all ideal network sources in the current design are displayed. If you specify a list of source pins or ports, the command displays information about the ideal networks emanating from these objects.

The "Latency" and "Transition" columns show the minimum and maximum values set for both rising and falling edges. Set these values with the **set_ideal_latency** and **set_ideal_transition** commands.

Multicorner-Multimode Support

This command applies only to the current scenario by default, but **-modes**, **-corners**, **-scenarios** lists are accepted.

EXAMPLES

The following example sets up an ideal network, applies ideal latency and ideal transition values and shows the output of the ideal network report.

```
prompt> set_ideal_network {p_in in1}
prompt> set_ideal_latency    -min  1.12 in1
prompt> set_ideal_transition -max  9.87 in1
prompt> set_ideal_latency    -min  1.34 n0_i/Z
prompt> set_ideal_transition -rise 5.62 n3_i/B
prompt> report_ideal_network -timing -load_pin -cell -net

****************************************
Report : ideal_network
      -timing
      -load_pin
      -net
      -cell
 ...
****************************************
```

```
Source ports                Latency                  Transition
and pins              Rise         Fall         Rise         Fall
                   min   max    min   max    min   max    min   max
-------------------------------------------------------------------------------
middle/in1        1.12   --    1.12   --     --   9.87    --   9.87
middle/p_in         --   --     --    --     --    --     --    --

Internal pins               Latency                  Transition
with ideal timing     Rise         Fall         Rise         Fall
                   min   max    min   max    min   max    min   max
-------------------------------------------------------------------------------
n3_i/B              --   --     --    --    5.62  5.62    --    --
n0_i/Z            1.34   --    1.34   --     --    --     --    --

Boundary pins               Latency                  Transition
                     Rise         Fall         Rise         Fall
                   min   max    min   max    min   max    min   max
-------------------------------------------------------------------------------
o_reg3/D            --   --     --    --     --    --     --    --
middle/p_out        --   --     --    --     --    --     --    --

Nets
-------------------------------------------------------------------------------
in1
p_in
p_out
n2
n1
n0

Cells
-------------------------------------------------------------------------------
n3_i
n2_i
n1_i
n0_i
```

The following example shows the output of the ideal network report on specified objects.

```
prompt> report_ideal_network -timing -load_pin -cell -net {in1}

****************************************
Report : ideal_network
        -timing
        -load_pin
        -net
        -cell
 ...
****************************************

Source ports                Latency                  Transition
and pins              Rise         Fall         Rise         Fall
                   min   max    min   max    min   max    min   max
-------------------------------------------------------------------------------
middle/in1        1.12   --    1.12   --     --   9.87    --   9.87
```

```
Boundary pins              Latency                  Transition
                    Rise         Fall         Rise         Fall
                  min   max    min   max    min   max    min   max
-------------------------------------------------------------------------------
o_reg3/D           --    --     --    --     --    --     --    --

Nets
-------------------------------------------------------------------------------
in1
```

SEE ALSO

remove_ideal_network(2)
report_constraint(2)
set_ideal_latency(2)
set_ideal_network(2)
set_ideal_transition(2)

report_ieee_1500_configuration

Displays the options specified by the **set_ieee_1500_configuration** command.

SYNTAX

status **report_ieee_1500_configuration**

ARGUMENTS

The **report_ieee_1500_configuration** command has no arguments.

DESCRIPTION

This command displays options applied by the **set_ieee_1500_configuration** command to the current design.

EXAMPLES

The following is an example of the report generated by the **report_ieee_1500_configuration** command:

```
prompt> report_ieee_1500_configuration
****************************************
Report : IEEE 1500 Configuration
Design : top
Version: P-2019.03
Date   : Thu Feb  7 07:01:03 2019
****************************************

IEEE 1500 Structures              Status
------------------              --------
WIR Width:                         1
WBY Width:                         1
```

SEE ALSO

set_ieee_1500_configuration(2)

report_ignored_layers

Reports the routing layers that are ignored during congestion analysis and RC estimation. If you have set the minimum and maximum routing layers for the design, they are also reported.

SYNTAX

status **report_ignored_layers**
[-verbose]
[-nosplit]
[-significant_digits *digits*]

Data Types

digits int

ARGUMENTS

-verbose

Displays verbose ignored layer information.

-nosplit

Does not split lines if column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default. Fixed-precision floating-point arithmetics is used for delay calculation; therefore, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h. The upper bound on a meaningful value of the argument to **-significant_digits** is then FLT_DIG - ceil(log10(fabs(*any_reported_value*))).

DESCRIPTION

During congestion analysis and RC estimation, the tool can ignore some routing layers. Use this command to report the ignored layers. If you have specified the minimum and maximum routing layers for the design, these are also reported.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the **report_ignored_layers** command:

```
prompt> report_ignored_layers
```

SEE ALSO

remove_ignored_layers(2)
report_lib(2)
set_ignored_layers(2)

report_incomplete_upf

Displays information about the original UPF errors and actions incomplete_upf support has handled.

SYNTAX

status **report_incomplete_upf**
[-max_message_count *message_count*]

Data Types

message_count integer

ARGUMENTS

-max_message_count *message_count*

Set the maximum number of occurrences for each message type. If you set it to zero or 'all', that means the number of occurrences of the message is unlimited. If this option is not specified, the limit is 20 by default.

DESCRIPTION

The **report_incomplete_upf** command displays information about original UPF errors and actions incomplete_upf support has handled.

EXAMPLE

The following command reports the original UPF errors and actions incomplete_upf support has handled:

```
prompt> report_incomplete_upf
report_incomplete_upf
****************************************
Report : report_incomplete_upf
Design : TOP
Version: L-2016.03-SP5
Date   : Wed Sep  7 11:41:25 2016
****************************************
```

Rule: MV-042 (P/G type)

Count	Action	Message
1	Ignore warning	Unable to derive power and ground type for supply net vdd_test

Rule: UPF-030 (Inconsistent resolution type)

Count	Action	Message

Rule: UPF-037 (Port state conflict)

Count	Action	Message

Rule: UPF-056 (Power states conflict)

Count	Action	Message

Rule: UPF-061 (No valid element for strategy)

Count	Action	Message

Rule: UPF-069 (pst_state conflict)

Count	Action	Message

Rule: UPF-083 (Strategy conflict)

Count	Action	Message

Rule: UPF-144 (Merge domain error)

Count	Action	Message

Rule: UPF-170 (Terminal boundary)

Count	Action	Message

Rule: UPF-203 (Bias block conflict)

```
---------------------------------------------------------------
Count  Action              Message
---------------------------------------------------------------

Rule: UPF-209 (Not supported options for bias)
---------------------------------------------------------------
Count  Action              Message
---------------------------------------------------------------

Rule: Missing connect_supply_net
---------------------------------------------------------------
Count  Action
  Message
---------------------------------------------------------------
1    Infer CSN (infer_from_primary)
 Missing connect_supply_net for pin 'U_0/o1_UPF_ISO/VDD'
 Inferred UPF: connect_supply_net VDD_H2 -ports U_0/o1_UPF_ISO/VDD

Summary
---------------------------------------------------------------
Rule        Total Message Count
---------------------------------------------------------------
MV-042       1
UPF-030      0
UPF-037      0
UPF-056      0
UPF-061      0
UPF-069      0
UPF-083      0
UPF-144      0
UPF-170      0
UPF-203      0
UPF-209      0
Missing CSN    1
1
```

SEE ALSO

load_upf(2)
commit_upf(2)
set_app_option(2)

report_individual_pin_constraints

Reports existing pin constraints on individual pins or nets.

SYNTAX

string **report_individual_pin_constraints**
[-pins *pins*]
[-ports *ports*]
[-nets *nets*]
[-constraint_type feedthrough | physical]

Data Types

pins	collection
ports	collection
nets	collection

ARGUMENTS

-pins *pin_collection*

Limits the constraint report to only the specified pins. By default, the command reports constraints on all pins.

-ports *port_collection*

Limits the constraint report to only the specified ports. By default, the command reports constraints on all ports.

-nets *net_collection*

Limits the constraint report to only the specified nets. By default, the command reports constraints on all nets.

-constraint_type *feedthrough* | *physical*

Limits the report to the specified constraint type. By default, the command reports both feedthrough and physical constraints.

DESCRIPTION

This command reports the individual pin constraints on every level. Also it reports order constraint associated with that pin constraints.

If you do not specify any options, the command reports the feedthroughs and physical pin constraints for all pins and nets.

EXAMPLES

The following example displays the pin physical constraints for pin I_ALU/Optr.

```
prompt> report_individual_pin_constraints -constraint_type physical \
  -pins [get_pins I_ALU/Optr]
```

SEE ALSO

remove_individual_pin_constraints(2)
set_individual_pin_constraints(2)

report_io_guides

Reports I/O guides in the current design.

SYNTAX

int **report_io_guides**
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*io_guide_list*]

Data Types

digits int
io_guide_list list

ARGUMENTS

-verbose

Displays additional information about the I/O guides.

-nosplit

Retains long lines and does not split lines if the column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option to override the default.

io_guide_list

Specifies a list of I/O guides to report. The list can contain I/O guide names, patterns, or collections. A collection can be specified by using the **get_io_guides** command.

DESCRIPTION

The **report_io_guides** command generates a report about the I/O guide constraints in the design. The report includes the I/O guide name and a description of the I/O guide. If **-verbose** is specified, the report also includes I/O guide line coordinates, associated side, whether the spacing is origin or neighbor relative, and pad cells with offset/spacing contained within the I/O guide.

If *io_guide_list* is specified, those I/O guides are reported. If it is not specified, then all I/O guides in the design are reported.

EXAMPLES

The following example reports all I/O guides in the design.

```
prompt> report_io_guides *
Io_guide        Description
-------------------------------------------------------------------------
ioGuide1        left with 4 pads origin_relative
ioGuide2        left with 8 pads neighbor_relative
ioGuide3        top with 8 pads neighbor_relative
ioGuide4        right with 8 pads neighbor_relative
ioGuide5        bottom with 8 pads neighbor_relative
1
```

The following example reports verbose information for the I/O guide named "ioGuide2".

```
prompt> report_io_guides -verbose ioGuide2*
Io_guide        Description
-------------------------------------------------------------------------
ioGuide2        left with 3 pads origin_relative
                line: ((0, 0) (0, 1000))
                pads:  pad1   5 / 100
                       pad2   3 / 200
                       pad3  -1 / 400
1
```

SEE ALSO

add_to_io_guide(2)
create_io_guide(2)
get_io_guides(2)
remove_from_io_guide(2)
remove_io_guides(2)
shell.common.report_default_significant_digits(3)

report_io_rings

Reports I/O rings in the current design.

SYNTAX

int **report_io_rings**
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*io_ring_list*]

Data Types

digits int
io_ring_list list

ARGUMENTS

-verbose

Displays verbose I/O ring information.

-nosplit

Does not split lines if column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default.

io_ring_list

Specifies a list of I/O rings to report. The list can contain I/O ring names, patterns, or collections. A collection can be specified by using the **get_io_rings** command.

DESCRIPTION

The **report_io_rings** command generates a report about the I/O ring constraints in the design. The report includes the I/O ring name and a description of the I/O ring. If **-verbose** is specified, the report also includes I/O ring related (outer) ring, inter-ring/boundary offset, and details of the guides.

If *io_ring_list* is specified, those I/O rings are reported. If it is not specified, then all I/O rings in the design are reported.

EXAMPLES

The following example reports I/O rings named "io_ring1".

```
prompt> report_io_rings { io_ring1 }
Io_ring         Description
-------------------------------------------------------------------------
io_ring1        io_guides: io_ring1.left
                   side: left
                   line: {{100 900} {100 100}}
                   number of pad cells: 0

                   io_ring1.bottom
                   side: bottom
                   line: {{100 100} {900 100}}
                   number of pad cells: 0

                   io_ring1.right
                   side: right
                   line: {{900 100} {900 900}}
                   number of pad cells: 0

                   io_ring1.top
                   side: top
                   line: {{900 900} {100 900}}
                   number of pad cells: 0
```

SEE ALSO

add_to_io_ring(2)
create_io_ring(2)
get_io_rings(2)
remove_from_io_ring(2)
remove_io_rings(2)
shell.common.report_default_significant_digits(3)

report_isolate_ports

Reports the input and output ports enabled for port isolation in the current block.

SYNTAX

string **report_isolate_ports**
[-all]
[*port_list*]

Data Types

port_list collection

ARGUMENTS

-all

Reports all the ports that are enabled for port isolation.

This option is mutually exclusive with the *port_list* argument.

port_list

Reports whether the specified ports are enabled for port isolation.

This argument is mutually exclusive with the **-all** option.

DESCRIPTION

This command reports the ports that are enabled for port isolation and the reason why port isolation is enabled. Currently the command reports only those ports enabled for port isolation with the **set_isolate_ports** command, which is indicated by the **user** reason.

Note that the command does not report the buffers added to the ports to implement the port isolation.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example enables port isolation on all ports for a block that has two ports, port1 and port2.

```
prompt> set_isolate_ports [get_ports *]
prompt> report_isolate_ports -all
Report : isolate_ports

Port port1
 user

Port port2
 user

2 of 2 ports are isolate_ports
1
```

The following example enables port isolation only on the port1 port.

```
prompt> set_isolate_ports [get_ports port1]
prompt> report_isolate_ports -all
Report : isolate_ports

Port port1
 user

1 of 2 ports are isolate_ports
1

prompt> report_isolate_ports [get_ports port2]
Report : isolate_ports

Port port2 is not isolate ports

0 of 1 ports are isolate_ports
1

prompt> report_isolate_ports [get_ports port1]
Report : isolate_ports

Port port1
 user

1 of 1 ports are isolate_ports
1
```

SEE ALSO

set_isolate_ports(2)

report_ivm

Reports via variation for current desgin.

SYNTAX

status **report_ivm**
[-corners *corners*]

Data Types

corners list

ARGUMENTS

-corners *corners*

Specifies the list of corners for which to apply the extraction options. If corners are not specified then the ivm file will be applied to all corners.

DESCRIPTION

This command reports via variation content sfor the current design.

EXAMPLES

The following example reports via variation for specified two corners.

```
prompt> report_ivm -corners {c1}
Report IVM Corner : c1
Layer Name: *
area          : 1.000000
table_min_sigma : 0.280000
table_max_sigma : 0.420000
table_meanshift : 0.000000
table_stdev     : 0.000000

Layer Name: VIA1
```

```
area          : 3.000000 4.000000 5.000000 6.000000
table_min_sigma : 0.700000 0.700000 0.700000 0.700000
table_max_sigma : 1.300000 1.300000 1.300000 1.300000
table_meanshift : 0.000000 0.000000 0.000000 0.000000
table_stdev     : 0.000000 0.000000 0.000000 0.000000

Layer Name: VIA2
area          : 3.000000 4.000000 5.000000 6.000000
table_min_sigma : 0.700000 0.700000 0.700000 0.700000
table_max_sigma : 1.300000 1.300000 1.300000 1.300000
table_meanshift : 0.000000 0.000000 0.000000 0.000000
table_stdev     : 0.000000 0.000000 0.000000 0.000000
```

SEE ALSO

read_ivm(2)
remove_ivm(2)
write_ivm(2)
write_parasitics(2)

report_keepout_margins

Reports the keepout margins

SYNTAX

status **report_keepout_margin**
[keepout_list]

Data Types

keepout_list Collection

ARGUMENTS

keepout_list

Specifies the keepout margin object to be reported. It could be a single object or a collection of keepout margins.

DESCRIPTION

The command reports the keepout margins on designs, partitions and cells. If the *keepout_list* is not specified, then the keepout margins of the current design are reported.

EXAMPLES

The following example reports all the keepouts in the current design:

prompt> **report_keepout_margin**

The following example reports all the keepouts associated with the cell 'U280':

prompt> **report_keepout_margin [get_keepout_margins -of {U280}]**

SEE ALSO

create_keepout_margin(2)
get_keepout_margins(2)
remove_keepout_margins(2)

report_latch_loop_groups

Reports on latch data pins involved in loops of transparent latches.

SYNTAX

string **report_latch_loop_groups**
[-of_objects *pin_list*]
[-loop_breakers_only]
[-path_breakers_only]
[-nosplit]

Data Types

pin_list list

ARGUMENTS

-of_objects *pin_list*

Specifies a collection of data pins of transparent latches. Only pins of loop groups containing the specified pins are reported. By default, the command reports on all latch data pins involved with transparent latch loops, as well as other latch data pins outside loops with the **is_latch_loop_breaker** pin attribute set to **true**.

-loop_breakers_only

Specifies that the report should contain only pins that are loop breaker latch data pins and should be pins truly contained in transparent latch loops. By default, all transparent latch data pins within the loop group are reported, along with other latch data pins with the **is_latch_loop_breaker** pin attribute set to **true**.

-path_breakers_only

Specifies that the report should contain only pins with the **is_latch_loop_breaker** pin attribute set to **true**. By default, all transparent latch data pins within the loop group are returned in the collections.

-nosplit

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing software to extract information from the report output.

DESCRIPTION

When sequential loops of transparent latches exist in a design, a loop group containing all the latch data pins of intersecting loops is formed. The **report_latch_loop_groups** command analyzes the design and reports on latch data pins involved with loops. The report also includes some latch data pins outside loops. Combinational pins are not included in the results.

Latch data pins outside loops can be reported if the latch is being treated by the tool as a latch loop breaker data pin, even if it is not inside a latch loop. This can happen when either the user requests the pin to be treated this way using the **set_latch_loop_breaker** command, or the tool has selected the pin to be treated as a loop breaker data pin to save runtime. For more information about how this can happen, see the man page of the **time.through_path_max_segments** application option.

The report lists the name of the latch data pin, a number identifying which latch loop group the pin is part of, and attributes describing if the pin is a loop breaker latch, and why.

The number identifying which latch loop group the pin is part of is an arbitrary number, except that within a report, the latch data pins that are part of the same loop group have the same number. For latch data pins that are not part of a loop group, "NA" is shown instead of a loop group number.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example uses the **report_latch_loop_groups** command to report latch loops in the design:

```
prompt> report_latch_loop_groups
****************************************
Report : latch loop groups
 ...
****************************************
Attributes
  b - loop breaker d pin
  s - self loop d pin
  p - long path breaker d pin, but not in a loop
  u - user requested to be a loop breaker using set_latch_loop_breaker
  a - user requested to avoid with set_latch_loop_breaker -avoid

 Latch                  Latch Loop   Attributes
 D pin                  Group
-------------------------------------------------------------
DUT/Latch1/D                NA          pu
DUT/Latch2/D                1           b
DUT/Latch3/D                1
DUT/Latch4/D                2           bu
DUT/Latch5/D                2           a
```

SEE ALSO

get_latch_loop_groups(2)
set_latch_loop_breaker(2)

report_lib

Displays information about the specified library, which can be either a design library or a reference library.

SYNTAX

status **report_lib**
[-nosplit]
[-timing_arcs]
[-parasitic_tech]
[-physical]
[-antenna]
[-routability]
[-min_pin_layer layer_name]
[-pattern_must_join_pin]
[-pattern_must_join_pin_exclusion_list lib_pin_list]
[-placement_constraints]
[-wire_tracks]
[-wire_track_colors]
[-technology_lib library_name]
[-verbose]
[-include_db_mapping]
[-cell_summary]
[-user_defined_cell_attributes cell_attribute_list]
[-user_defined_pin_attributes pin_attribute_list]
[-char_model]
library

Data Types

library collection
library_name Library name in simple, relative, or absolute path.
layer_name string
lib_pin_list collection of lib_pins
cell_attribute_list collection of cell attributes
pin_attribute_list collection of pin attributes

ARGUMENTS

-nosplit

Most of the library information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

-timing_arcs

Reports all cell timing arcs from the library.

-parasitic_tech

Reports the parasitic technology information from the library.

-physical

Reports the physical characteristics of the library.

-antenna

Reports missing antenna properties of cells only in frame view in the specified library.

-routability

Reports routability related items.

-min_pin_layer

Specifies the minimum layer pin layer name. Pins on layers lower than this layer will be reported. This option needs to go with **-routability**.

-pattern_must_join_pin

Report pattern must-join related items.

-pattern_must_join_pin_exclusion_list

Specifies the collection of lib_pins that don't need the pattern must-join report. This option needs to go with **-pattern_must_join_pin**.

-placement_constraints

Reports placement related constraints status.

-wire_tracks

Reports the via regions that are on wire tracks.

-wire_track_colors

Reports the terminals that are on improper color tracks.

-technology_lib

Specifies the technology library in which the wire track information is stored. This option needs to go with **-wire_tracks** or **-color_wire_tracks**.

-verbose

Set the detail reporting mode. This option is for **-routability**, **-placement_constraints, -wire_tracks and -color_wire_tracks only.**

-include_db_mapping

Includes the DB mapping information in the report.

-cell_summary

Reports a Library Cell Summary.

-user_defined_cell_attributes

Specifies cell level user defined attributes to be appended to Library Cell Summary. This option needs to go with **-cell_summary**.

-user_defined_pin_attributes

Specifies pin level user defined attributes to be appended to Library Cell Summary. This option needs to go with **-cell_summary**.

-char_model

Reports Data Model Existence information.

library

Specifies the reference library to report. This can be either the library name or a collection of a single library.

DESCRIPTION

The **report_lib** command displays information about the specified design library or reference library. By default, the command reports the full name, file name, and timing data, if any, for the library. To report the associated parasitic technology files, use the **-parasitic_tech** option.

In general, design libraries will not have timing data, but reference libraries will. The default timing data includes the power rails, the details for each timing pane, the constituent logic libraries, and the operating conditions. You can also include the cell timing arcs by using the **-timing_arcs** option.

The **-physical** option reports information about sites, layers, the path, source files, cell details and pin details in the library.

The **-antenna** option will display antenna property report table which lists the names of the cells only in frame view and mostly 10 pins in module/macro/pad cells at once to be printed that are missing an antenna property and shows the missing property type.

The specified library must be loaded into memory. To display the libraries that are currently loaded in memory, use the **get_libs** command.

To report routability related items which will affect routing runtime and QoR, use the **-routability** option. These items include:

- Physical pin accessibility for via region and access edge.
  Missing via region or access edge might cause routing QoR issue. The cells which miss via region or access edge are listed in the report. Please note that for a port of which all the physical pins are deep pins and are accessible from upper layer, then this port is count as accessible even if there are no access edges on all physical pins.
  - Design type for checking via region: diode, end cap, fill, filler, lib cell, physical only, well tap
  - Design type for checking access edge: abstract, analog, block box, corner, cover, flip-chip driver, flip-chip pad, macro, module, pad, pad spacer
- The total number of obstructions exceed 100K which will cause long runtime issue during routing.
- Irregular cut shapes or irregular routing blockages on the cut layers.
  A cut layer shape or regular routing blockage which does not meet any of the following conditions is considered as irregular:
  - is not rectangular
  - the width and height attributes does not match any of the following values
    - the minimum width of the layer
    - the default width of the layer

    - the values in cutHeightTbl and cutWidthTbl of the layer
    - the values of cut size of any simple via definitions of the layer

  These shapes or blockages may cause unpredictable design rule violations and long runtime issue during routing.

- Improper frame options setting
  For the following design type: diode, end cap, fill, filler, lib cell, physical only, well tap
    - The value of 'trim_metal_blockage_around_pin' option should not be set as 'all'.
    - The value of 'block_all' option should not be set as 'true'
- The terminal short violations.
  The physical geometry of ports should not short with others.
- Routing port without geometry.
  All routing ports should have physical geometry except the device-bias ports or internal ports.
- Summary table of cell objects.
  Please use the **-verbose** option together.
- Terminals blocked by blockages or metal shapes.
  The terminals should not be blocked by any blockages or metal shapes.
- Inconsistent routing attributes between timing view and frame view
  If the attribute 'is_secondary_pg' or 'is_diode' is defined in timing view, then it should be also defined in frame view.
- Improper internal port settings.
  The ports with attributes 'direction = internal' and 'is_physical = false' will not be treated as a real metal by router. Thus such settings is reported as improper internal port settings.
- Terminals on layers lower than minimum pin layer.
  The terminals might be on layers lower than minimum pin layer, for example M0. The minimum pin layer is given by the option **-min_pin_layer**.
- Terminals, shapes, and regular routing blockages with end of preferred grid issue.
  Terminals, shapes, or regular routing blockages which end in between low end grid and high end grid will cause issue when router tries to do metal extension. The end of preferred grid need to be pre-defined by command **create_track_pattern -end_grid_low_offset {VALUE} -end_grid_high_offset {VALUE} -end_grid_low_steps {VALUE} -end_grid_high_steps {VALUE}**. Different end of preferred grid patterns can be created for different site definitions, and the actual position of the low end grids and high end grids will be calculated using the patterns of the cell associated site definition. Use **create_track_pattern -site SITE** option to create the patterns for non-default site definitions.

To report the incorrect pattern-must-join setting on lib_pins, use the **-pattern_must_join_pin** option. Only the pattern-must-join settings saved in frame can be checked when reporting. Design-level lib_pin attribute override value is not considered for the report. For skipping the report for specific lib_pins, please specify them using the option **-pattern_must_join_pin_exclusion_list**.

To report the missing geometry on tech-file defined device layers and improper site definition, use the **-placement_constraints** option. Tech-file defined device layers includes implant, diffusion, and so on. Missing geometry on these layers might cause placement QoR issue.

- Design type for checking geometry: diode, end cap, fill, filler, lib cell, physical only, well tap

To report the via regions which are not on wire tracks, use the **-wire_tracks** option. If the **-technology_lib** is specified, then tool will get the wire track information from the specified technology library. Otherwise, the wire track information will come from the target library. If there are no wire track information set in the library, then tool calculates the wire tracks by the pitch, offset, and routing direction attributes on layer. For reporting the detail via region on wire tracks status, use the **-verbose** option together.

To report the terminals which are on improper color wire tracks, use the **-wire_track_colors** option. The color tracks need to be pre-defined by command **create_track_pattern -mask_pattern {PATTERN} -site SITE**. Different patterns can be created for different

site definitions, and the color tracks will be calculated using the patterns of the cell associated site definition. If the **-technology_lib** is specified, then tool will get the wire track information from the specified technology library. Otherwise, the wire track information will come from the target library. This part of report is available only in the library manager.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following command generates a default report for the my_lib_fast reference library.

```
prompt> report_lib -include_db_mapping my_lib_fast

****************************************
Report : library
 ...

****************************************

Full name: /usr/my_lib_fast.nlib:my_lib_fast
File name: /usr/my_lib_fast.nlib
Design count: 527
Timing data:

Power rails:

Index  Name       Type
 0    <default>   power

Pane count: 2
Pane  0:
  Process label: slow
  Process number: 1.00
  Voltage rail count: 1
  Voltage for rail 0 (<default>): 1.08
  Temperature: 125

  Thresholds:
    r/f InputDelay: 0.5/0.5  r/f OutputDelay: 0.5/0.5
    l/h RiseTrans:  0.1/0.9  h/l FallTrans:   0.9/0.1
    TransDerate: 1

Pane  1:
  Process label: fast
  Process number: 1.00
  Voltage rail count: 1
  Voltage for rail 0 (<default>): 1.32
  Temperature: -40

  Thresholds:
    r/f InputDelay: 0.5/0.5  r/f OutputDelay: 0.5/0.5
    l/h RiseTrans:  0.1/0.9  h/l FallTrans:   0.9/0.1
```

```
    TransDerate: 1

  Constituent .db libraries:

   my_fast_lib_125.db:my_fast_lib_125
   my_fast_lib_n40.db:my_fast_lib_n40

  Operating Conditions:
   Name Process Temp   Voltage Original DB Name  Original DB Filename
    Mapped DB Filename
   ------------------------------------------------------------------
   slow 1.00    125.00 1.08    my_fast_lib_125   my_fast_lib_125.db
    (none)

   fast 1.00    -40.00 1.32    my_fast_lib_n40   my_fast_lib_n40.db
    (none)

  1
```

The following example uses -include_db_mapping -routability and -min_pin_layer to report routability related issue in the current opened library.

```
prompt> report_lib [current_lib] -include_db_mapping -routability -min_pin_layer M1

****************************************
Report : library
Library: mylib_lvt_pg
Version: M-2016.12-SP2
Date   : Tue Jan 17 21:33:10 2017
****************************************

Full name: /usr/mylib_lvt_pg.nlib:mylib_lvt_pg
File name: /usr/mylib_lvt_pg.nlib
Design count: 20

--------------------------------------------------------------------------------
** Timing Data:

Power rails:

Index  Name       Type
 0    <default>   power
 1    VDD         power
 2    VDDG        power
 3    VSS        ground
 4    VSSG        ground

Pane count: 2

Pane 0:
 Process label: (none)
 Process number: 1.01
 Voltage rail count: 3
 Voltage for rail 0 (<default>): 0.95
 Voltage for rail 1 (VDD): 0.95
 Voltage for rail 2 (VDDG): 0.95
 Temperature: 125
```

```
  Thresholds:
    r/f InputDelay: 0.5/0.5  r/f OutputDelay: 0.5/0.5
    l/h RiseTrans:  0.2/0.8  h/l FallTrans:   0.8/0.2
    TransDerate:    1

  Source .db file:
    /usr/LIBRARIES/DB/mylib_lvt_pg.db



Pane 1:
  Process label: (none)
  Process number: 1.01
  Voltage rail count: 3
  Voltage for rail 0 (<default>): 0.95
  Voltage for rail 1 (VDD): 0.95
  Voltage for rail 2 (VDDG): 0.95
  Temperature: -40

  Thresholds:
    r/f InputDelay: 0.5/0.5  r/f OutputDelay: 0.5/0.5
    l/h RiseTrans:  0.2/0.8  h/l FallTrans:   0.8/0.2
    TransDerate:    1

  Source .db file:
    /usr/LIBRARIES/DB/mylib_lvt_pg_1.db


Source .db libraries:
  mylib_lvt.db:mylib_lvt_pg
  mylib_lvt_1.db:mylib_lvt_pg_1


Operating Conditions:

  Name              Process   Label             Temp      Voltage
    Original DB Name     Original DB Filename
  ------------------------------------------------------------------------
  pg                1.01      (none)            125.00    0.95
    mylib_lvt_pg         mylib_lvt_pg.db
  pg_1              1.01      (none)            -40.00    0.95
    mylib_lvt_pg_1       mylib_lvt_pg_1.db


#BEGIN_REPORT_ROUTABILITY

--------------------------------------------------------------------------------
Target Library: mylib_lvt_pg
--------------------------------------------------------------------------------


Report of missing pin access:
Total number of cells without optimal routability: 0 (out of 20)

Report of irregular cut shapes or routing blockages:
```

```
Total number of cells with irregular cut layer shapes or routing blockages:
  0 (out of 20)


Report of terminal overlapped:
Total number of cells with terminal overlapped: 0 (out of 20)

Report of blocked terminals:
Total number of cells with blocked terminals: 0 (out of 20)

Report of routing ports without geometry:
Total number of cells with routing ports without geometry: 0 (out of 20)

Report of ports with inconsistent routing attributes:
Total number of cells with ports with inconsistent routing attributes:
  0 (out of 20)

Report of improper internal port settings:
Total number of cells with improper internal port settings: 0 (out of 20)

Report of terminals on layers lower than minimum pin layer:
Warning: The terminal 'SLEEP/SLEEP' of cell 'FOOT_LVT' is on layer 'M0'
  lower than the minimum pin layer. (FRAM-055)
Total number of cells with terminals on layers lower than minimum pin layer:
  1 (out of 20)

#END_REPORT_ROUTABILITY
1
```

SEE ALSO

create_lib(2)
open_lib(2)
current_lib(2)
get_libs(2)
set_db_file_mapping(2)
create_track_pattern(2)

report_lib_cells

Reports information about lib_cell objects

SYNTAX

status **report_lib_cells**
-objects *objects*]
[-columns *column_specs*]

Data Types

objects list
column_specs list

ARGUMENTS

-objects *objects*

Specifies the objects to include in the report. Each object will be printed as a separate row. For this command, each object must be of class "lib_cell".

-columns *column_specs*

Specifies the columns to include in the report, their order, and optionally column width and number of significant digits for each column. Each entry is a column spec, which can take the following forms. If just a column name is specified, such as "area", it will use report-specific default column width and significant digits. A column spec such as "area:10" means to use a column width of 10. A column spec such as "area:10.4" means to use a column width of 10 and 4 significant digits. The significant digits information only applies to floating point values. If any column spec is "all", all valid columns will be included for this report.

The following column names are valid for this report: area dont_touch full_name design_type is_combinational is_isolation is_level_shifter is_memory_cell is_sequential is_three_state name pad_cell pin_count valid_purposes view_name

DESCRIPTION

This command prints a tabular report with information on a list of lib_cell objects.

EXAMPLES

The following example prints a report for all lib_cells in library "tech_lib", using the default report configuration.

```
prompt> report_lib_cells -objects [get_lib_cells -of_objects tech_lib]
```

This example reports on all lib_cells matching "XOR*" in library "tech_lib", with a specific column configuration.

```
prompt> report_lib_cells -objects [get_lib_cells tech_lib/XOR*] \
 -columns {full_name:20 area:10.4 pin_count}
```

SEE ALSO

set_report_configuration(2)

report_lib_pins

Reports information about lib_pin objects

SYNTAX

status **report_lib_pins**
-objects *objects*]
[-columns *column_specs*]

Data Types

objects list
column_specs list

ARGUMENTS

-objects *objects*

Specifies the objects to include in the report. Each object will be printed as a separate row. For this command, each object must be of class "lib_pin".

-columns *column_specs*

Specifies the columns to include in the report, their order, and optionally column width and number of significant digits for each column. Each entry is a column spec, which can take the following forms. If just a column name is specified, such as "full_name", it will use report-specific default column width and significant digits. A column spec such as "full_name:10" means to use a column width of 10. A column spec such as "length:10.4" means to use a column width of 10 and 4 significant digits. The significant digits information only applies to floating point values. If any column spec is "all", all valid columns will be included for this report.

The following column names are valid for this report: direction disable_timing full_name function is_async_pin is_clock_pin is_data_pin is_pad is_preset_pin is_three_state is_three_state_enable_pin is_three_state_output_pin name pin_number port_type rail_name signal_type three_state_function

If **-columns** is not specified, the report will use the current columns specification which is either the tool default for this report, or the user-specified value from **set_report_configuration**.

DESCRIPTION

This command prints a tabular report with information on a list of lib_pin objects.

EXAMPLES

The following example prints a report for all lib_pins in library "tech_lib", using the default report configuration.

```
prompt> report_lib_pins -objects [get_lib_pins tech_lib/*/*]
```

This example reports on all lib_pins matching "XOR*/Y" in library "tech_lib", with a specific column configuration.

```
prompt> report_lib_pins -objects [get_lib_pins tech_lib/XOR*/Y] \
 -columns {name:10 direction function}
```

SEE ALSO

set_report_configuration(2)

report_lib_timing_arcs

Reports information about lib_timing_arc objects

SYNTAX

status **report_lib_timing_arcs**
-objects *objects*]
[-columns *column_specs*]

Data Types

objects list
column_specs list

ARGUMENTS

-objects *objects*

Specifies the objects to include in the report. Each object will be printed as a separate row. For this command, each object must be of class "lib_timing_arc".

-columns *column_specs*

Specifies the columns to include in the report, their order, and optionally column width and number of significant digits for each column. Each entry is a column spec, which can take the following forms. If just a column name is specified, such as "sdf_cond", it will use report-specific default column width and significant digits. A column spec such as "sdf_cond:10" means to use a column width of 10. A column spec such as "length:10.4" means to use a column width of 10 and 4 significant digits. The significant digits information only applies to floating point values. If any column spec is "all", all valid columns will be included for this report.

The following column names are valid for this report: direction disable_timing full_name function is_async_pin is_clock_pin is_data_pin is_pad is_preset_pin is_three_state is_three_state_enable_pin is_three_state_output_pin name pin_number port_type rail_name signal_type three_state_function

If **-columns** is not specified, the report will use the current columns specification which is either the tool default for this report, or the user-specified value from **set_report_configuration**.

DESCRIPTION

This command prints a tabular report with information on a list of lib_timing_arc objects.

EXAMPLES

The following example prints a report for all lib_timing_arcs in library "tech_lib", using the current configuration for this report.

```
prompt> report_lib_timing_arcs -objects [get_lib_timing_arcs -of_objects tech_lib/*/*]
```

This example reports on all lib_timing_arcs on lib_cell "XOR2" in library "tech_lib", with a specific column configuration.

```
prompt> report_lib_timing_arcs -objects [get_lib_timing_arcs -of_objects tech_lib/XOR2] \
 -columns {from:10 to:10 sense:15 when is_disabled }
```

SEE ALSO

set_report_configuration(2)

report_libcell_subset

Reports the library cell subset family specified on sequential and instantiated combinational cells.

SYNTAX

status **report_libcell_subset**
[-object_list *cells*]
[-nosplit]

Data Types

cells list

ARGUMENTS

-object_list *cells*

Specifies the cells for which to report the library cell subset family.

-nosplit

Specifies not to split lines when column fields overflow.

DESCRIPTION

This command reports information about the library cell subset subset that is specified on the cells.

Note that the command only reports the library subset information that is explicitly user-defined.

EXAMPLES

The following example reports the library cell subset family for the u1 sequential cell:

```
prompt> report_libcell_subset -object_list [get_cells u1]

Libcell Family     Libcells
-------------------------------------------------------------------------------
special_flops      sff0 sff1 sff2
```

```
--------------------------------------------------------------------------------
Object          Reference    Libcell Family
--------------------------------------------------------------------------------
u1              sff0         special_flops
--------------------------------------------------------------------------------
1
```

SEE ALSO

remove_libcell_subset(2)
report_libcell_subset(2)
set_libcell_subset(2)
define_libcell_subset(2)

report_logic_levels

Displays logic levels information about a design.

SYNTAX

status **report_logic_levels**
[-from *from_list*
| -rise_from *rise_from_list*
| -fall_from *fall_from_list*]
[-to *to_list*]
| -rise_to *rise_to_list*
| -fall_to *fall_to_list*]
[-through *through_list*]
[-rise_through *rise_through_list*]
[-fall_through *fall_through_list*]
[-exclude *exclude_list*
| -rise_exclude *rise_exclude_list*
| -fall_exclude *fall_exclude_list*]
[-delay_type max | min | max_rise | max_fall | min_rise | min_fall]
[-path_type *path_type*]
[-sort_paths_by *sort_path_value*]
[-nworst *paths_per_endpoint*]
[-max_paths *max_path_count*]
[-num_bins *number_of_bins*]
[-bin_width *bin_width_number*]
[-max_paths_to_report *max_path_report_count*]
[-slack_lesser_than *lesser_slack_limit*]
[-slack_greater_than *greater_slack_limit*]
[-start_end_pair]
[-export_csv_file *output_csv_file_name*]
[-summary_only]
[-nosplit]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-groups *group_list*]

Data Types

from_list list
rise_from_list list
fall_from_list list
to_list list
rise_to_list list
fall_to_list list
through_list list
rise_through_list list

fall_through_list list
exclude_list list
rise_exclude_list list
fall_exclude_list list
paths_per_endpoint integer
max_path_count integer
number_of_bins integer
bin_width_number integer
max_path_report_count integer
lesser_slack_limit float
greater_slack_limit float
path_type string
sort_path_value string
mode_list list
corner_list list
scenario_list list
group_list list

ARGUMENTS

-from *from_list*

Reports only the paths that originate from the named pins, ports, cells, or clocks. If you do not specify this option, the command reports the longest path to an output port if the design has no timing constraints. If the design has timing constraints, the command reports the path with the worst slack within each path group if **-group** is not present. If **-group** is given, the command reports the path with the worst slack within the group specified by **-group**.

-rise_from *rise_from_list*

Reports only the paths with a rising edge from the named pins, ports, cells, or clocks. If a clock object is specified, this option selects startpoints clocked by the named clock, if the paths are launched by the rising edge of the clock at the clock source. The command considers logical inversions along the clock path.

-fall_from *fall_from_list*

Reports only the paths with a falling edge from the named pins, ports, cells, or clocks. If a clock object is specified, this option selects startpoints clocked by the named clock, if the paths are launched by the falling edge of the clock at the clock source. The command considers logical inversions along the clock path.

-to *to_list*

Reports only the paths that connect to the named pins, ports, cells, or clocks. If you do not specify this option, the command reports the longest path to an output port if the design has no timing constraints. If the design has timing constraints, the command reports the path with the worst slack within each path group if **-group** is not present. If **-group** is given, the command reports the path with the worst slack within the group specified by **-group**.

-rise_to *rise_to_list*

Reports only the paths with a rising edge at the endpoint that connect to the named pins, ports, cells, or clocks. If a clock object is specified, this option selects endpoints clocked by the named clock, if the capturing edge is the rising edge of the clock at the clock source. The command considers logical inversions along the clock path.

-fall_to *fall_to_list*

Reports only the paths with a falling edge at the endpoint that connect to the named pins, ports, cells, or clocks. If a clock object is

specified, this option selects endpoints clocked by the named clock, if the capturing edge is the falling edge of the clock at the clock source. The command considers logical inversions along the clock path.

-through ***through_list***

Reports only paths that pass through the named pins, ports, cells, or nets. If you specify only one **-through** option, the tool reports only the paths that travel through one or more of the objects in the list. You can specify **-through** multiple times with one **report_logic_levels** command.

-rise_through ***rise_through_list***

Reports only paths that pass through the named pins, ports, cells, or nets and have a rising transition at those objects. If you specify only one **-through** option, the tool reports only the paths that travel through one or more of the objects in the list. You can specify **-rise_through** multiple times with one **report_logic_levels** command.

-fall_through ***fall_through_list***

Reports only paths that pass through the named pins, ports, cells, or nets and have a falling transition at those objects. If you specify only one **-through** option, the tool reports only the paths that travel through one or more of the objects in the list. You can specify **-fall_through** multiple times with one **report_logic_levels** command.

-exclude ***exclude_list***

Prevents the reporting of data paths that contain the specified pins, ports, nets, and cell instances. Reporting excludes all data paths from, through, or to the named pins, ports, nets, and cell instances. If a cell instance is specified, all pins of the cell are excluded. The **-exclude** option

o Takes precedence over the **-from**, **-through**, and **-to** options.

o Is exclusive with the **-rise_exclude** and **-fall_exclude** options.

o Does not apply to clock pins.

Note that this option is intended for use in conjunction with **-from**, **-through** and **-to** options and can lead to long runtime when used on its own, i.e. without other such topological options.

-rise_exclude ***rise_exclude_list***

Same as the **-exclude** option, but applies only to paths rising at the named pins, ports, nets, and cell instances.

-fall_exclude ***fall_exclude_list***

Same as the **-exclude** option, but applies only to paths falling at the named pins, ports, nets, and cell instances.

-delay_type max | **min** | **max_rise** | **max_fall** | **min_rise** | **min_fall**

Specifies the type of path delay. Valid values are **max** (the default), **min**, **max_rise**, **max_fall**, **min_rise**, or **min_fall**. The "rise" or "fall" suffix in the delay type refers to a rising or falling transition at the path endpoint.

-path_type all | **single_clock** | **cross_clock** | **multicycle_paths** | **infeasible_paths**

Specifies the type of path to be considered for logic level analysis. By default, the all paths are considered.

-sort_paths_by wns | **logic_level_violation** | **logic_level**

Specifies the criterion on which to sort the paths which will be displayed in the logic level path report section. Only the top paths specified by option -max_paths_to_report will be reported after sorting.

-nworst ***paths_per_endpoint***

Specifies the maximum number of paths to get per endpoint. The default is 1, which gets only the single worst path ending at a given endpoint.

-max_paths ***max_path_count***

Specifies the maximum number of paths to analyze per path group. The default is 100.

-num_bins number_of_bins

Specifies the number of bins to be used in the Logic level dis- tribution section. If this option is not specified, the tool uses default value of 10.

-bin_width bin_width_number

Specifies the width of the bin to be used in the Logic level distribution section. If both **-num_bins** & **-bin_width** options are specified, then only **-num_bins** is used. When **-bin_width** option is specified, the tool computes the number of bins based on this width value.

-max_paths_to_report max_path_report_count

Specifies the number of paths to be reported in the logic level path report section.

-slack_lesser_than ***lesser_slack_limit***

Selects only those paths with a slack less than lesser_slack_limit. The **-slack_greater_than option** can be com- bined with the **-slack_lesser_than** option to select only those paths inside or outside of a given slack range. The specified value is in main library units. The default value for this option is 0.0. Be default, the command will analyze only the -ve slack paths due to this default setting.

-slack_greater_than ***greater_slack_limit***

Selects only those paths with a slack greater than **greater_slack_limit**. The **-slack_greater_than** option can be com- bined with the **-slack_lesser_than** option to select only those paths inside or outside of a given slack range. The specified value is in main library units.

-start_end_pair

Reports paths for each pair of startpoint and endpoint based on connectivity. This option can lead to long runtime and can lead to generating a huge number of paths depending on the design. By default, this option searches only for paths that are violating. This default can be changed by having an explicit -slack_lesser_than option. The options that do not work with this option are -nworst and -max_paths.

-export_csv_file output_csv_file_name

Specifies the file name for the csv file. The entire contents of the report will be exported to the specified csv file in csv (comma separated values) format.

-summary_only

If this option is specified, then only the summary section is printed. Logic level distribution section & Logic level path report for each of the path groups are not reported.

-nosplit

Writes out wide report lines, even if the line length is greater tha 80 columns. Use this option to simplify post-processing of the output files or when comparing the output files with the UNIX **diff** command. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-modes ***mode_list***

Specifies the modes for which to generate the report. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is reported separately, with up to **-max_paths** paths reported for each path group of each scenario. If this option is not given, the command reports scenarios of all modes. If none of the **-modes**, **-corners**, or **-**

scenarios options are given, the command reports every active scenario of the design. It is an error to give this option with the **-scenarios** option.

-corners ***corner_list***

Specifies the corners for which to generate the report. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is reported separately, with up to **-max_paths** paths reported for each path group of each scenario. If this option is not given, the command reports every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios ***scenario_list***

Specifies the scenarios for which to generate the report. Each of the specified scenarios is reported. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command will report every scenario of the design.

-groups ***group_list***

Specify the list of path group names for reporting paths. Default is all path groups.

DESCRIPTION

This command reports the logic levels information about the design.

By default, the command analyses only -ve slack paths for each scenarios from all path groups with a maximum of 100 paths for each path group.

The command collects the paths according to the user specifications and calculates the logic levels in those paths. Logic levels are the number of mapped cells in the path excluding buffers and invertors. It then compares it against the threshold specified by the user before or auto- matically computed one and determines whether it violates the threshold or not. It also collects other information about the path such as WNS, start and end point, start and end clocks and reports them.

The report has three sections.

Summary section

In this section, logic level and timing information are reported
for cumulative and each of the path groups. The cumulative
information is reported as "All path groups".
The columns in this section can be customized using
shell.synthesis.logic_level_report_summary_format of **set_app_options**.

GROUP	REQUIRED PERIOD	WNS	NUM OF PATHS	MAX LEVEL
All Path Groups	n/a	n/a	2310	13
CLOCK	0.0100	-2.3289	1600	13
custom_group1	0.0100	-2.3174	166	12
custom_group2	0.0100	-2.3006	544	13

Summary section is followed by Logic level distribution section & Logic level path report section for cumulative and each of the path groups. The cumulative information is reported under the path group name "All path groups".

Logic level distribution section

In this section, logic level distribution of paths for the particular path group are reported. The columns in this section are not customizable and are fixed.

LOGIC LEVELS	NUMBER OF PATHS	PERCENTAGE OF PATHS
0 to 1	4	0%
2 to 3	22	1%
4 to 5	1017	44%
6 to 7	908	39%
8 to 9	47	2%
10 to 11	205	9%
12 to 13	107	5%

Logic level path report

In this section, logic level & timing information for the top paths for the particular path group are reported. The columns in this section are customizable through setting an option **shell.synthesis.logic_level_report_group_format** using **set_app_options**.

WNS	LOGIC LEVELS	THRESHOLD	START POINT	END POINT
-1.3106	13	4	REG_BLK/DATA_OUT_reg[0]/CK	UPC_BLK/DATA_OUT_reg[7]/D
-1.3072	11	4	REG_BLK/DATA_OUT_reg[0]/CK	UPC_BLK/DATA_OUT_reg[3]/D
-1.2999	10	4	REG_BLK/DATA_OUT_reg[0]/CK	UPC_BLK/DATA_OUT_reg[3]/D
-1.2991	11	4	REG_BLK/DATA_OUT_reg[0]/CK	UPC_BLK/DATA_OUT_reg[9]/D
-1.2979	12	4	REG_BLK/DATA_OUT_reg[0]/CK	UPC_BLK/DATA_OUT_reg[5]/D

EXAMPLES

The following example sets the global logic level threshold to 5 and then runs logic level report for path group "CLOCK"

```
prompt> set_analyze_rtl_logic_level_threshold 5
prompt> report_logic_levels -group CLOCK
```

The following example invokes the reporting command with csv file option.

```
prompt> report_logic_levels -export_csv_file  ll.csv
```

SEE ALSO

get_timing_paths(2)
set_app_options(3)

report_logicbist_configuration

Displays options specified by the **set_logicbist_configuration** command.

SYNTAX

status **report_logicbist_configuration**

ARGUMENTS

The **report_logicbist_command** has no arguments.

DESCRIPTION

This command displays options specified by the **set_logicbist_configuration** command on the current design.

EXAMPLES

The following is an example of a LogicBIST compression configuration report:

```
prompt> report_logicbist_configuration
****************************************
Report : Logic Bist Configuration
Design : top
Version: P-2019.03
Date   : Thu Feb  7 07:06:30 2019
****************************************

Maximum Chain Length              8
Pattern counter width            12
Clock Name                      CLK
----------------------------------------
TEST MODE: selftest
VIEW    : Specification
----------------------------------------
```

SEE ALSO

set_logicbist_configuration(2)

report_macro_constraints

Reports macro placement constraints.

SYNTAX

status **report_macro_constraints**
[-allowed_orientations]
[-preferred_location]
[-alignment_grid]
[-align_pins_to_tracks]
[*hard_macro_list*]

Data Types

hard_macro_list list of hard macros

ARGUMENTS

-allowed_orientations

Reports allowed orientation constraints and legal orientation only.

-preferred_location

Reports preferred macro location constraints only.

-alignment_grid

Reports alignment grid and alignment point information only.

-align_pins_to_tracks

Reports the align_pins_to_tracks setting only.

hard_macro_list

Specifies the list of hard macros for which report constraint settings. By default, all hard macros are reported.

DESCRIPTION

This command reports on various options set by the **set_macro_constraints** command. You can use any combination of the **-allowed_orientations**, **-preferred_location -align_pins_to_tracks**, **-align_pins_to_tracks** options to generate the report. The

command also reports the legal macro orientations from the reference library.

EXAMPLES

The following example reports the macro constraint settings for all hard macros.

```
prompt> report_macro_constraints
****************************************
Report : report_macro_constraints
 ...

****************************************

Macro               allowed       legal
Name                orientations   orientations
-------------------------------------------------------
macro1
                R0            R0 MX
macro2
                R0 R90        all
macro3
                R0 R90 MX MY   all

Macro       R0 grid   R0 alignment  R90 grid   R90 alignment
name        name     point       name      point
-------------------------------------------------------------------------
macro1
          grid1    {3 7}       grid1
macro2
          grid1               grid2     {10 15}


Macro       R0 align pins         R90 align pins
Name        to tracks             to tracks
-------------------------------------------------------------------------
macro1
          false               true
macro2
          false               false
macro3
          false               false
```

SEE ALSO

create_placement(2)
set_macro_constraints(2)

report_macro_relative_location

Reports the previously set constraints which place hard macros or macro array relative to anchor objects.

SYNTAX

status **report_macro_relative_location**
[*target_list*]
[-hierarchical]
[-nosplit]

Data Types

target_list list

ARGUMENTS

target_list

Specifies the list of hard macros or macro arrays on which to report relative location constraints. By default is reporting constraints on all hard macros and macro arrays.

-hierarchical

Report macro relative location constraints and violation checking results in current block and child block. Default only report constraints in current block.

-nosplit

Report macro relative location constraints and violation checking results with nosplit line format. Default will split the constraint's object name if it is too long.

DESCRIPTION

This command reports hard macro or macro array relative location constraints set by the **set_macro_relative_location** command.

EXAMPLES

The following command reports the relative location constraint set on hard macro A1.

```
prompt> report_macro_relative_location [get_cells "A1"]
```

SEE ALSO

set_macro_relative_location(2)
remove_macro_relative_location(2)
derive_macro_relative_location(2)
write_macro_relative_location(2)
create_placement(2)
create_macro_relative_location_placement(2)

report_matching_types

Reports matching_types in the current block.

SYNTAX

int **report_matching_types**
[-nosplit]
[*matching_type_list*]

Data Types

matching_type_list list

ARGUMENTS

-nosplit

Does not split lines if column overflows.

matching_type_list

Specifies a list of matching types to report. The list may contain matching type names, patterns, or collections. A collection may be specified by using the **get_matching_types** command.

DESCRIPTION

The **report_matching_types** command generates a report about the matching types in the design. The report includes the matching type name, its uniquify number, and its list of containing objects.

If *matching_type_list* is specified, those matching types will be reported. If it is not specified, then all matching types in the block will be reported.

EXAMPLES

The following example reports all matching types in the design.

prompt> **report_matching_types** *

1

The following example reports verbose information for the matching type named "myMatchingType1".

```
prompt> report_matching_types myMatchingType1
```

1

SEE ALSO

create_matching_type(2)
get_matching_types(2)
add_to_matching_type(2)
remove_from_matching_type(2)
remove_matching_types(2)

report_mibs

Generates a report of the multiply instantiated blocks (MIBs) for physical block cells in the design.

SYNTAX

string **report_mibs**
  [-ignore_read_only_mibs]

ARGUMENTS

-ignore_read_only_mibs

Reports multiply instantiated blocks (MIBs) for physical block cells in the design. You can specify the **-ignore_read_only_mibs** options. By default, the command reports all multiple instantiated blocks(MIBs) for physical block cells in the design. If **-ignore_read_only_mibs** options are specified, then the command will skips read only blocks.

DESCRIPTION

This command write out a report for all multiple instantiated blocks (MIBs) for physical block cells in the design. This report describes which references have multiple instances plus additional information on their status such as view name, design name and placement status, editability, and so on. The orientation reported for the physical block cells is with respect to the top block. The command reports all MIBs across every level of physical hierarchy in the design.

EXAMPLES

The following example reports MIBs in the design

```
prompt> report_mibs
report_mibs
****************************************
Report : report_mibs
 ...

****************************************

Reference   View   Hierarchy  Count  Instance      Orientation  Status  Editability
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
```

```
BLENDER_0   design Block    5    I_ORCA_TOP/I_BLENDER_1 R0   Placed  True
                    I_ORCA_TOP/I_BLENDER_6 R0   Placed  True
                    I_ORCA_TOP/I_BLENDER_5 R0   Placed  True
                    I_ORCA_TOP/I_BLENDER_4 R0   Placed  True
                    I_ORCA_TOP/I_BLENDER_3 R0   Placed  True
BLENDER_6   design Block    2    I_ORCA_TOP/I_BLENDER_7 R0   Placed  True
                    I_ORCA_TOP/I_BLENDER_2 R0   Placed  True
```

SEE ALSO

change_reference(2)
commit_block(2)
get_mib_objects(2)
uniquify(2)

report_min_period

Displays minimum period check information about specified sequential device clock pins.

SYNTAX

status **report_min_period**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-path_type *path_type*]
[-all_violators]
[-significant_digits *digits*]
[-nosplit]
[*port_pin_list*]
[-input_pins]
[-derate]
[-capacitance]
[-transition_time]
[-variation]
[-crosstalk_delta]
[-nets]

Data Types

mode_list collection
corner_list collection
scenario_list collection
path_type collection
digits int
port_pin_list list

ARGUMENTS

-modes *mode_list*

Specifies the modes to be reported. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is reported separately. If this option is not given, the command reports scenarios of the current mode. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners to be reported. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is reported separately. If this option is not given, the command reports scenarios of the current corner. If

none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any). It is an error to give this option with the **-scenarios** option.

-scenarios ***scenario_list***

Specifies the scenarios to be reported. Each of the given active scenarios is reported. It is an error to give this option with the **-modes** or **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any).

-all_violators

Reports violated min period only.

-path_type

Specifies the format of the path report and how the clock path is displayed. Allowed values are: summary(the default), which generates a report with a column format that shows one line for each path and shows only the required period, actual period and slack. short, which displays only start-points and endpoints in the clock path.full_clock, which displays full clock paths. full_clock_expanded, which displays full clock paths including all master clocks of a generated clock.

-significant_digits ***digits***

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13. the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default.

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line splitting and facilitates writing software to extract information from the report output.

port_pin_list

Specifies a list of pins to report. If no pin is given, by default, the command reports all clock pins in the current design. If no min_pulse_width constraints found on pins or they are not on clock network, they will be ignored.

-input_pins

Reports input pins of cell as well as output pins.

-derate

Indicates that derate factors are shown in the path report. The default is to show no derate factors. This also shows the derate factor specified on the required min_period if one has been specified by the set_timing_derate command. Note that this option applies only when the -path_type option is set to either full_clock or full_clock_expanded options.

-variation

Includes parametric on-chip variation (POCV) information in the report in columns labeled "Mean" and "Sensit". POCV analysis is enabled by setting the timer.pocvm_enable_analysis variable to true.

-nets

Reports nets. The default is not to show nets.

-crosstalk_delta

Reports the annotated delta delay and delta transition time. The deltas are computed during crosstalk signal integrity analysi. Note that only deltas on input pins are shown. Delta transition time is shown only with the -transition_time option. The -crosstalk_delta option automatically sets the -input_pins option.

-transition_time

Reports the transition time (slew). The default is not to show time. For each driver pin or load pin the transition time is displayed in a column preceding incremental path delay.

-capacitance

Reports the total (lump) capacitance. The default is not to show capacitance. When the -nets option is specified, the capacitance is printed on the lines with nets instead of the lines with driver pins. Note that the -capacitance option is used with -nets option.

DESCRIPTION

The **report_min_period** command displays the following information for the minimum period check: required minimum period, actual period, and by how much the constraint is violated or met (slack). This information is listed in order of decreasing slack starting with the worst violator. This command displays information for the current scenario.

This command will trigger additional timing update and may have some runtime penalty.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example generates a report of all minimum period violated or met in the current design.

```
prompt> report_min_period
```

****
Report : min_period -path_type summary -significant_digits
Design : Decoder
Version: Q-2019.12-ALPHA
Date : Thu Oct 10 16:25:50 2019
****
Mode: default
Corner: default
Scenario: default
Required Actual Pin min period period Slack
---
ram/CK (fall) 4.50 2.50 -2.00 (VIOLATED)
ram/CK (rise) 4.47 2.50 -1.97 (VIOLATED)

The following example generates a expanded full path report. It reports minimum period violated or met of port 'ram/CK' in the current design, including propagation starts from master clocks.

```
prompt> report_min_period ram/CK -path_type full_clock_expanded -significant_digits 4
```

****
Report : min_period -path_type full_clock_expanded -significant_digits
Design : Decoder
Version: Q-2019.12-ALPHA
Date : Thu Oct 17 11:16:52 2019
****
Mode: default
Corner: default
Scenario: default
Pin: ram/CK
Related clock: g1clk1
Point Incr Path
---
clock g1clk1'(fall edge) 2.5000 2.5000
source latency 0.0000 2.5000
clk1 (in) 0.0000 2.5000 f
io_clk1/IO (BD2SCRDQP_40) 0.0000 2.5000 f
io_clk1/ZI (BD2SCRDQP_40) 0.2523 2.7524 f
clk1iv1/A (IVLLP) 0.1075 2.8599 f
clk1iv1/Z (IVLLP) 0.4178 3.2777 r
clk1iv1/Z (IVLLP) (gclock 'g1clk1' source) 0.0000 3.2777 r
clk1iv3/A (IVLLP) 0.0027 3.2804 r
clk1iv3/Z (IVLLP) 0.7670 4.0474 f
ram/CK (SPLARGE_6144x32m12) 0.1540 4.2014 f
open edge clock latency 4.2014
clock g1clk1'(fall edge) 7.5000 7.5000
source latency 0.0000 7.5000
clk1 (in) 0.0000 7.5000 f
io_clk1/IO (BD2SCRDQP_40) 0.0000 7.5000 f
io_clk1/ZI (BD2SCRDQP_40) 0.2205 7.7205 f
clk1iv1/A (IVLLP) 0.0940 7.8145 f
clk1iv1/Z (IVLLP) 0.3650 8.1795 r
clk1iv1/Z (IVLLP) (gclock 'g1clk1' source) 0.0000 8.1795 r
clk1iv3/A (IVLLP) 0.0024 8.1819 r
clk1iv3/Z (IVLLP) 0.6702 8.8521 f
ram/CK (SPLARGE_6144x32m12) 0.1346 8.9867 f

duty cycle clock jitter -0.5000 8.4867 clock uncertainty -0.8000 7.6867 close edge clock latency 7.6867 ---------- open edge clock latency 4.2014 close edge clock latency 7.6867 ---------- actual period 3.4853 ---------- required min period 4.4979 actual period 3.4853 ---------- slack (VIOLATED) -1.0126 Pin: ram/CK Related clock: g1clk1 Point Incr Path ---------- clock g1clk1'(rise edge) 0.0000 0.0000 source latency 0.0000 0.0000 clk1 (in) 0.0000 0.0000 r io_clk1/IO (BD2SCRDQP_40) 0.0000 0.0000 r io_clk1/ZI (BD2SCRDQP_40) 0.2735 0.2736 r clk1iv1/A (IVLLP) 0.1152 0.3888 r clk1iv1/Z (IVLLP) 0.4183 0.8071 f clk1iv1/Z (IVLLP) (gclock 'g1clk1' source) 0.0000 0.8071 f clk1iv3/A (IVLLP) 0.0022 0.8093 f clk1iv3/Z (IVLLP) 0.8890 1.6983 r ram/CK (SPLARGE_6144x32m12) 0.1552 1.8535 r open edge clock latency 1.8535 clock g1clk1'(rise edge) 5.0000 5.0000 source latency 0.0000 5.0000 clk1 (in) 0.0000 5.0000 r io_clk1/IO (BD2SCRDQP_40) 0.0000 5.0000 r io_clk1/ZI (BD2SCRDQP_40) 0.2390 5.2390 r clk1iv1/A (IVLLP) 0.1007 5.3397 r clk1iv1/Z (IVLLP) 0.3655 5.7052 f clk1iv1/Z (IVLLP) (gclock 'g1clk1' source) 0.0000 5.7052 f clk1iv3/A (IVLLP) 0.0019 5.7071 f clk1iv3/Z (IVLLP) 0.7768 6.4840 r ram/CK (SPLARGE_6144x32m12) 0.1356 6.6196 r duty cycle clock jitter -0.5000 6.1196 clock uncertainty -0.8000 5.3196 close edge clock latency 5.3196 ---------- open edge clock latency 1.8535 close edge clock latency 5.3196 ---------- actual period 3.4661 ---------- required min period 4.4654 actual period 3.4661 ---------- slack (VIOLATED) -0.9993

SEE ALSO

report_min_pulse_width(2)
shell.common.report_default_significant_digits(3)
timer.pocvm_enable_analysis(3)
.

report_min_pulse_width

Displays minimum pulse width check information about specified sequential device clock pins.

SYNTAX

status **report_min_pulse_width**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-all_violators]
[-significant_digits *digits*]
[-nosplit]
[*pin_list*]

Data Types

mode_list collection
corner_list collection
scenario_list collection
digits int
pin_list list

ARGUMENTS

-modes *mode_list*

Specifies the modes to be reported. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is reported separately. If this option is not given, the command reports scenarios of the current mode. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners to be reported. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is reported separately. If this option is not given, the command reports scenarios of the current corner. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios to be reported. Each of the given active scenarios is reported. It is an error to give this option with the **-modes** or **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the command uses the current scenario (if any).

-all_violators

Reports violated min pulse width only.

-significant_digits *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default.

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line splitting and facilitates writing software to extract information from the report output.

pin_list

Specifies a list of pins to report. If no pin is given, by default, the command reports all clock pins in the current design. If no min_pulse_width constraints found on pins or they are not on clock network, they will be ignored.

DESCRIPTION

The **report_min_pulse_width** command displays the following information for the minimum pulse width check: required pulse width, actual pulse width, and by how much the constraint is violated or met (slack). This information is listed in order of decreasing slack starting with the worst violator. This command displays information for the current scenario.

Minimum pulse width violations can also be reported with the **report_constraint** command.

This command will trigger additional timing update and may have some runtime penalty.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example generates a report of all minimum pulse width violations in the current design.

prompt> **report_min_pulse_width**

****
Report : min_pulse_width
Design : test
Version: F-2011.09-SP
Date : Tue Oct 18 13:24:39 2011
****
Mode: scen.mode
Corner: scene.corner
Scenario: scen
Required Actual Pin pulse width pulse width Slack
----
ff1/cp(low) 2.00 0.87 -1.13 (VIOLATED)
ff2/cp(low) 2.00 1.41 -0.59 (VIOLATED)
ff2/cp(high) 0.60 0.59 -0.01 (VIOLATED)
ff1/cp(high) 0.60 1.13 0.53 (MET)

SEE ALSO

remove_min_pulse_width(2)
set_min_pulse_width(2)
shell.common.report_default_significant_digits(3)

report_mismatch_configs

Reports the available mismatch configs.

SYNTAX

status **report_mismatch_configs**
[-config_list *config_names*]
[-all]
[-nosplit]

Data Types

config_names list

ARGUMENTS

-config_list

List of the config names to be reported.

-all

Reports detailed information about all the existing configs.

-nosplit

Does not split lines if column overflows.

DESCRIPTION

The **report_mismatch_configs** command reports the named config. If no config is passed, then this command gives detailed information of current config of the session. If option -all is given, this command reports detailed information about all the configs, including pre-defined as well as user-defined configs.

EXAMPLES

The following example reports the detailed information about current config:

```
prompt> report_mismatch_configs
****************************************
Report : Reporting Mismatch Configs
 ...

****************************************


Config : default
--------------------------------------------------------------------------------
Mismatch-Type                  Action  Allowed Actions
  Repair-Strategy
    Available Strategies
--------------------------------------------------------------------------------
lib_missing_physical_reference    error   {repair error accept }
  null
    {create_placeholder_physical_lib_cell }
lib_missing_logical_port         error   {repair error accept }
  null
    {create_placeholder_logic_lib_port }
lib_missing_physical_port        error   {repair error accept }
  null
    {create_placeholder_physical_lib_port }
lib_port_type                  accept  {accept }
  change_lib_port_type
    {change_lib_port_type }
lib_port_direction             accept  {accept }
  change_lib_port_direction
    {change_lib_port_direction }
lib_port_name_synonym           ignore  {repair accept ignore }
  null
    {record_lib_port_name_synonym }
missing_logical_reference        error   {repair error accept }
  record_unbound_instance
    {create_blackbox record_unbound_instance }
missing_port                   error   {error accept }
  null
    {delete_extra_port }
port_name_case                 error   {error accept }
  record_port_name_case_insensitivity
    {record_port_name_case_insensitivity }
reference_name_case             error   {repair error accept }
  record_cell_name_case_insensitivity
    {record_cell_name_case_insensitivity }
port_name_synonym              error   {error accept }
  record_port_name_synonym
    {record_port_name_synonym }
lib_cell_name_case             ignore  {repair accept ignore }
  null
    {record_lib_cell_name_case_insensitivity }
lib_port_name_case             ignore  {repair accept ignore }
  null
    {record_lib_port_name_case_insensitivity }
lib_missing_rail_pg            ignore  {repair accept ignore }
  null
    {create_pg_for_rail }
```

```
bus_bit_naming                error   {error accept }
  bus_bit_blast_naming
    {bus_bit_blast_naming }
layer_missing_prefer_direction    repair  {repair accept }
  assign_prefer_direction_for_metal_layer
    {assign_prefer_direction_for_metal_layer }
lib_port_missing_via_region       ignore  {ignore }
  record_the_port_missing_via_region
    {record_the_port_missing_via_region }
lib_port_missing_access_edge      ignore  {ignore }
  record_the_port_missing_access_edge
    {record_the_port_missing_access_edge }
macro_cell_contain_too_many_obs   ignore  {ignore }
  record_the_macro_cell_contains_too_many_obstructions
    {record_the_macro_cell_contains_too_many_obstructions }
exceed_ndr_limitation             repair  {repair accept }
  delete_ndr_rules_exceed_limitation
    {delete_ndr_rules_exceed_limitation }
empty_logic_module                error   {repair accept ignore error }
  null
    {create_macro }

1
```

The following example shows detailed information about config user_def:

```
prompt> report_mismatch_configs -config_list user_def
****************************************
Report : Reporting Mismatch Configs
 ...

****************************************


Config : user_def
--------------------------------------------------------------------------------
Mismatch-Type                 Action  Allowed Actions
  Repair-Strategy
    Available Strategies
--------------------------------------------------------------------------------
lib_missing_physical_reference   error   {repair error accept }
  null
    {create_placeholder_physical_lib_cell }
lib_missing_logical_port         error   {repair error accept }
  null
    {create_placeholder_logic_lib_port }
lib_missing_physical_port        error   {repair error accept }
  null
    {create_placeholder_physical_lib_port }
lib_port_type                 accept  {accept }
  change_lib_port_type
    {change_lib_port_type }
lib_port_direction            accept  {accept }
  change_lib_port_direction
    {change_lib_port_direction }
lib_port_name_synonym         ignore  {repair accept ignore }
  null
```

```
    {record_lib_port_name_synonym }
  missing_logical_reference       error   {repair error accept }
    record_unbound_instance
    {create_blackbox record_unbound_instance }
  missing_port                    error   {error accept }
    null
    {delete_extra_port }
  port_name_case                  error   {error accept }
    record_port_name_case_insensitivity
    {record_port_name_case_insensitivity }
  reference_name_case             error   {repair error accept }
    record_cell_name_case_insensitivity
    {record_cell_name_case_insensitivity }
  port_name_synonym               error   {error accept }
    record_port_name_synonym
    {record_port_name_synonym }
  lib_cell_name_case              ignore  {repair accept ignore }
    null
    {record_lib_cell_name_case_insensitivity }
  lib_port_name_case              ignore  {repair accept ignore }
    null
    {record_lib_port_name_case_insensitivity }
  lib_missing_rail_pg             ignore  {repair accept ignore }
    null
    {create_pg_for_rail }
  bus_bit_naming                  error   {error accept }
    bus_bit_blast_naming
    {bus_bit_blast_naming }
  layer_missing_prefer_direction  repair  {repair accept }
    assign_prefer_direction_for_metal_layer
    {assign_prefer_direction_for_metal_layer }
  lib_port_missing_via_region     ignore  {ignore }
    record_the_port_missing_via_region
    {record_the_port_missing_via_region }
  lib_port_missing_access_edge    ignore  {ignore }
    record_the_port_missing_access_edge
    {record_the_port_missing_access_edge }
  macro_cell_contain_too_many_obs ignore  {ignore }
    record_the_macro_cell_contains_too_many_obstructions
    {record_the_macro_cell_contains_too_many_obstructions }
  exceed_ndr_limitation           repair  {repair accept }
    delete_ndr_rules_exceed_limitation
    {delete_ndr_rules_exceed_limitation }
  empty_logic_module              error   {repair accept ignore error }
    null
    {create_macro }
  1
```

The following example shows detailed information about list of config names:

```
prompt> report_mismatch_configs -config_list {auto_fix default}
****************************************
Report : Reporting Mismatch Configs
 ...

****************************************
```

```
Config : auto_fix
--------------------------------------------------------------------------------
Mismatch-Type                 Action  Allowed Actions
  Repair-Strategy
    Available Strategies
--------------------------------------------------------------------------------
lib_missing_physical_reference   repair  {repair error accept }
  create_placeholder_physical_lib_cell
    {create_placeholder_physical_lib_cell }
lib_missing_logical_port        repair  {repair error accept }
  create_placeholder_logic_lib_port
    {create_placeholder_logic_lib_port }
lib_missing_physical_port       repair  {repair error accept }
  create_placeholder_physical_lib_port
    {create_placeholder_physical_lib_port }
lib_port_type                 accept  {accept }
  change_lib_port_type
    {change_lib_port_type }
lib_port_direction            accept  {accept }
  change_lib_port_direction
    {change_lib_port_direction }
lib_port_name_synonym           repair  {repair accept ignore }
  record_lib_port_name_synonym
    {record_lib_port_name_synonym }
missing_logical_reference       repair  {repair error accept }
  create_blackbox
    {create_blackbox record_unbound_instance }
missing_port                  accept  {error accept }
  delete_extra_port
    {delete_extra_port }
port_name_case                accept  {error accept }
  record_port_name_case_insensitivity
    {record_port_name_case_insensitivity }
reference_name_case           accept  {repair error accept }
  record_cell_name_case_insensitivity
    {record_cell_name_case_insensitivity }
port_name_synonym             accept  {error accept }
  record_port_name_synonym
    {record_port_name_synonym }
lib_cell_name_case            repair  {repair accept ignore }
  record_lib_cell_name_case_insensitivity
    {record_lib_cell_name_case_insensitivity }
lib_port_name_case            repair  {repair accept ignore }
  record_lib_port_name_case_insensitivity
    {record_lib_port_name_case_insensitivity }
lib_missing_rail_pg           repair  {repair accept ignore }
  create_pg_for_rail
    {create_pg_for_rail }
bus_bit_naming                accept  {error accept }
  bus_bit_blast_naming
    {bus_bit_blast_naming }
layer_missing_prefer_direction   accept  {repair accept }
  assign_prefer_direction_for_metal_layer
    {assign_prefer_direction_for_metal_layer }
lib_port_missing_via_region     ignore  {ignore }
```

```
  record_the_port_missing_via_region
    {record_the_port_missing_via_region }
lib_port_missing_access_edge    ignore  {ignore }
  record_the_port_missing_access_edge
    {record_the_port_missing_access_edge }
macro_cell_contain_too_many_obs  ignore  {ignore }
  record_the_macro_cell_contains_too_many_obstructions
    {record_the_macro_cell_contains_too_many_obstructions }
exceed_ndr_limitation          accept  {repair accept }
  delete_ndr_rules_exceed_limitation
    {delete_ndr_rules_exceed_limitation }
empty_logic_module             repair  {repair accept ignore error }
  create_macro
    {create_macro }


Config : default
--------------------------------------------------------------------------------
Mismatch-Type                  Action  Allowed Actions
  Repair-Strategy
    Available Strategies
--------------------------------------------------------------------------------
lib_missing_physical_reference   error   {repair error accept }
  null
    {create_placeholder_physical_lib_cell }
lib_missing_logical_port        error   {repair error accept }
  null
    {create_placeholder_logic_lib_port }
lib_missing_physical_port       error   {repair error accept }
  null
    {create_placeholder_physical_lib_port }
lib_port_type                  accept  {accept }
  change_lib_port_type
    {change_lib_port_type }
lib_port_direction             accept  {accept }
  change_lib_port_direction
    {change_lib_port_direction }
lib_port_name_synonym          ignore  {repair accept ignore }
  null
    {record_lib_port_name_synonym }
missing_logical_reference       error   {repair error accept }
  record_unbound_instance
    {create_blackbox record_unbound_instance }
missing_port                   error   {error accept }
  null
    {delete_extra_port }
port_name_case                 error   {error accept }
  record_port_name_case_insensitivity
    {record_port_name_case_insensitivity }
reference_name_case            error   {repair error accept }
  record_cell_name_case_insensitivity
    {record_cell_name_case_insensitivity }
port_name_synonym              error   {error accept }
  record_port_name_synonym
    {record_port_name_synonym }
lib_cell_name_case             ignore  {repair accept ignore }
```

```
    null
      {record_lib_cell_name_case_insensitivity }
  lib_port_name_case            ignore  {repair accept ignore }
    null
      {record_lib_port_name_case_insensitivity }
  lib_missing_rail_pg           ignore  {repair accept ignore }
    null
      {create_pg_for_rail }
  bus_bit_naming                error   {error accept }
    bus_bit_blast_naming
      {bus_bit_blast_naming }
  layer_missing_prefer_direction   repair  {repair accept }
    assign_prefer_direction_for_metal_layer
      {assign_prefer_direction_for_metal_layer }
  lib_port_missing_via_region     ignore  {ignore }
    record_the_port_missing_via_region
      {record_the_port_missing_via_region }
  lib_port_missing_access_edge    ignore  {ignore }
    record_the_port_missing_access_edge
      {record_the_port_missing_access_edge }
  macro_cell_contain_too_many_obs  ignore  {ignore }
    record_the_macro_cell_contains_too_many_obstructions
      {record_the_macro_cell_contains_too_many_obstructions }
  exceed_ndr_limitation         repair  {repair accept }
    delete_ndr_rules_exceed_limitation
      {delete_ndr_rules_exceed_limitation }
  empty_logic_module            error   {repair accept ignore error }
    null
      {create_macro }
  1
```

The following example shows detailed information about all available config:

```
prompt> report_mismatch_configs -all
****************************************
Report : Reporting Mismatch Configs
 ...

****************************************


Config : auto_fix
--------------------------------------------------------------------------------
Mismatch-Type                 Action  Allowed Actions
  Repair-Strategy
    Available Strategies
--------------------------------------------------------------------------------
lib_missing_physical_reference   repair  {repair error    accept }
  create_placeholder_physical_lib_cell
    {create_placeholder_physical_lib_cell }
lib_missing_logical_port       repair  {repair error    accept }
  create_placeholder_logic_lib_port
    {create_placeholder_logic_lib_port }
lib_missing_physical_port      repair  {repair error    accept }
  create_placeholder_physical_lib_port
    {create_placeholder_physical_lib_port }
lib_port_type                 accept  {accept }
```

```
  change_lib_port_type
    {change_lib_port_type }
lib_port_direction          accept  {accept }
  change_lib_port_direction
    {change_lib_port_direction }
lib_port_name_synonym       repair  {repair    accept    ignore }
  record_lib_port_name_synonym
    {record_lib_port_name_synonym }
missing_logical_reference   repair  {repair error    accept }
  create_blackbox
    {create_blackbox record_unbound_instance }
missing_port                accept  {error accept }
  delete_extra_port
    {delete_extra_port }
port_name_case              accept  {error accept }
  record_port_name_case_insensitivity
    {record_port_name_case_insensitivity }
reference_name_case         accept  {repair error accept }
  record_cell_name_case_insensitivity
    {record_cell_name_case_insensitivity }
port_name_synonym           accept  {error accept }
  record_port_name_synonym
    {record_port_name_synonym }
lib_cell_name_case          repair  {repair    accept    ignore }
  record_lib_cell_name_case_insensitivity
    {record_lib_cell_name_case_insensitivity }
lib_port_name_case          repair  {repair    accept    ignore }
  record_lib_port_name_case_insensitivity
    {record_lib_port_name_case_insensitivity }
lib_missing_rail_pg         repair  {repair    accept    ignore }
  create_pg_for_rail
    {create_pg_for_rail }
bus_bit_naming              accept  {error accept }
  bus_bit_blast_naming
    {bus_bit_blast_naming }
layer_missing_prefer_direction  accept  {repair accept }
  assign_prefer_direction_for_metal_layer
    {assign_prefer_direction_for_metal_layer }
lib_port_missing_via_region     ignore  {ignore }
  record_the_port_missing_via_region
    {record_the_port_missing_via_region }
lib_port_missing_access_edge    ignore  {ignore }
  record_the_port_missing_access_edge
    {record_the_port_missing_access_edge }
macro_cell_contain_too_many_obs ignore  {ignore }
  record_the_macro_cell_contains_too_many_obstructions
    {record_the_macro_cell_contains_too_many_obstructions }
exceed_ndr_limitation       accept  {repair accept }
  delete_ndr_rules_exceed_limitation
    {delete_ndr_rules_exceed_limitation }
empty_logic_module          repair  {repair accept ignore error }
  create_macro
    {create_macro }
```

Config : default

```
--------------------------------------------------------------------------------
Mismatch-Type                Action  Allowed Actions
  Repair-Strategy
    Available Strategies
--------------------------------------------------------------------------------
lib_missing_physical_reference   error   {repair error    accept }
  null
    {create_placeholder_physical_lib_cell }
lib_missing_logical_port       error   {repair error    accept }
  null
    {create_placeholder_logic_lib_port }
lib_missing_physical_port      error   {repair error    accept }
  null
    {create_placeholder_physical_lib_port }
lib_port_type                accept  {accept }
  change_lib_port_type
    {change_lib_port_type }
lib_port_direction           accept  {accept }
  change_lib_port_direction
    {change_lib_port_direction }
lib_port_name_synonym          ignore  {repair    accept ignore }
  null
    {record_lib_port_name_synonym }
missing_logical_reference      error   {repair error    accept }
  record_unbound_instance
    {create_blackbox record_unbound_instance }
missing_port                 error   {error    accept }
  null
    {delete_extra_port }
port_name_case               error   {error    accept }
  record_port_name_case_insensitivity
    {record_port_name_case_insensitivity }
reference_name_case          error   {repair error    accept }
  record_cell_name_case_insensitivity
    {record_cell_name_case_insensitivity }
port_name_synonym            error   {error    accept }
  record_port_name_synonym
    {record_port_name_synonym }
lib_cell_name_case           ignore  {repair    accept ignore }
  null
    {record_lib_cell_name_case_insensitivity }
lib_port_name_case           ignore  {repair    accept ignore }
  null
    {record_lib_port_name_case_insensitivity }
lib_missing_rail_pg          ignore  {repair    accept ignore }
  null
    {create_pg_for_rail }
bus_bit_naming               error   {error    accept }
  bus_bit_blast_naming
    {bus_bit_blast_naming }
layer_missing_prefer_direction   repair  {repair    accept }
  assign_prefer_direction_for_metal_layer
    {assign_prefer_direction_for_metal_layer }
lib_port_missing_via_region    ignore  {ignore }
  record_the_port_missing_via_region
    {record_the_port_missing_via_region }
```

```
lib_port_missing_access_edge     ignore  {ignore }
  record_the_port_missing_access_edge
    {record_the_port_missing_access_edge }
macro_cell_contain_too_many_obs  ignore  {ignore }
  record_the_macro_cell_contains_too_many_obstructions
    {record_the_macro_cell_contains_too_many_obstructions }
exceed_ndr_limitation          repair  {repair     accept }
  delete_ndr_rules_exceed_limitation
    {delete_ndr_rules_exceed_limitation }
empty_logic_module             error   {repair accept ignore error }
  null
    {create_macro }
1
```

SEE ALSO

report_design_mismatch(2)
get_mismatch_types(2)
get_mismatch_objects(2)
set_current_mismatch_config(2)
create_mismatch_config(2)
report_mismatch_configs(2)

report_missing_via_check_options

Reports missing via check options for the RedHawk Analysis Fusion check missing via analysis flow that were previously defined with the **set_missing_via_check_options** command.

SYNTAX

status **report_missing_via_check_options**

EXAMPLES

The following example reports the current settings for the missing via check.

prompt> **report_missing_via_check_options**

SEE ALSO

analyze_rail(2)
set_missing_via_check_options(2)
remove_missing_via_check_options(2)
set_app_options(2)

report_modes

Reports mode information.

SYNTAX

string **report_modes**
[-significant_digits *digits*]
[-nosplit]
  [*mode_list*]

list *mode_list*

ARGUMENTS

mode_list

Lists modes to report. If you do not specify this option, all modes in the current design are reported.

-significant_digits *digits*

Specifies the number of digits after the decimal point to be displayed for time values in the generated report. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, the tool uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

-nosplit

Writes out wide report lines, even if the line length is greater than 80 columns. Use this option to simplify post-processing of the output files or when comparing the output files with the UNIX **diff** command. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

Displays information about modes in the current design, including the corners that each mode is activated with.

Multicorner-Multimode Support

SEE ALSO

create_mode(2)
current_mode(2)

report_msg

Returns a list of matching messages

SYNTAX

integer **report_msg**
[msgTag]
[-level *msgLevel*]
[-format msgFormat]
[-printed]
[-triggered]
[-suppressed]
[-limit_set]
[-downgraded]
[-upgraded]
[-full_format]
[-silent]
[-filter]
[-summary]

Data Types

msgLevel string
msgTag string

ARGUMENTS

msgTag

Prints only messages of which the tag matches *msgTag* .

-level *msgLevel*

Prints only messages of which the level is msgLevel. Valid types are *fatal*, *serror*, *error*, *warning*, *info*, *verbose* or *off*.

-printed

Prints only messages that were printed since program start. This option is implicit in case no msgTag is supplied.

-triggered

Prints only messages that were triggered (encountered) since program start. Not all triggered messages are printed, but all printed messages are triggered.

-suppressed

Prints only messages that were printed fewer times than triggered.

-limit_set

Prints only messages that have a print limit set.

-downgraded

Prints only messages that are downgraded to a level that lower than their default level. Often these messages are not printed.

-upgraded

Prints only messages that were upgraded to a higher level than their default level. These are verbose messages that were made printable, messages that are upgraded to SEVERE_ERROR.

-full_format

Prints the entire format string, rather than a truncated one.

-silent

Prints nothing. Use this to have the command return the number of matching messages.

-filter

Prints the list of output filter strings currently active. (see set_msg -add_filter)

-summary

Returns the message summary report of the last command run

DESCRIPTION

The **report_message** command lists any message in the system. Messages are identified by their tag in the form 'ABC-1234' or 'ABC-123'.

EXAMPLES

The following example lists all messages that were printed:

prompt> **report_msg**

The following example lists all messages that match CMD:

prompt> **report_msg CMD***

The following example lists all suppressed messages:

prompt> **report_msg -suppressed**

SEE ALSO

set_msg(2)
get_msg(2)

report_multi_input_switching_coefficient

Reports delay coefficient for multi-input switching (MIS) analysis.

SYNTAX

int **report_multi_input_switching_coefficient**
[-nosplit]
[-significant_digits *num_digits*]
[*object_list*]

Data Types

num_digits integer
object_list list

ARGUMENTS

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13; the default is determined by the **shell.report_default_significant_digits** application option, whose default is 2. Use this option to override the default.

-nosplit

Retains long lines in the output, even if a column overflows. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

object_list

Specifies a list of library cells.

DESCRIPTION

This command reports delay coefficient for multi-input switching (MIS) analysis for the specified list of library cells. If you do not specify a list of library cells, the tool reports the coefficients for all library cells if set.

EXAMPLES

The following example reports all coefficients set by set_multi_input_switching_coefficient command:

```
pt_shell> report_multi_input_switching_coefficient
```

SEE ALSO

set_multi_input_switching_coefficient(2)
reset_multi_input_switching_coefficient(2)
enable_multi_input_switching_analysis(3)
enable_multi_input_switching_timing_window_filter(3)

report_multi_vth_constraint

Reports percentage vth enable/disable status, low vth percentage limit and cost type

SYNTAX

report_multi_vth_constraint

DESCRIPTION

This command reports percentage vth constraint, including enable/disable status, low vth percentage limit and cost type. These settings are persistent across shell sessions.

If percentage vth flow is enabled, optimization commands including **place_opt**, **clock_opt** and **route_opt** will honor the low vth percentage limit based on cost type. Low vth percentage limit is a floating-point number in range [0.0, 100.0]. Supported cost types are **cell_cout** and **area**.

EXAMPLES

```
prompt> report_multi_vth_constraint
Percentage vth flow  : disabled
Percentage vth limit : 100.000 (%)
Percentage vth cost  : cell_count
```

SEE ALSO

set_multi_vth_constraint(2)
reset_multi_vth_constraint(2)
report_threshold_voltage_group(2)
set_vth_threshold_controls(2)

report_multibit

Reports all multibit registers and multi-voltage cells in a design and the banking ratio.

SYNTAX

status **report_multibit**
[-nosplit]
[-hierarchical]
[-mv_cell iso | ls | els | all]
[-ignored_cells]

ARGUMENTS

-nosplit

Prevents line splitting. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-hierarchical

Reports all applicable multibit cells, the total number of cells, and the banking ratio in each of the submodules across the design hierarchy.

-mv_cell iso | **ls** | **els** | **all**

Specifies the multi-voltage cells to be included in the report. By default, these cells are not included, and only registers are reported.

A value of iso includes isolation cells; a value of ls includes level shifter cells; a value of els includes enable level shifter cells. A value of all includes isolation, level shifter, and enable level shifter cells.

Registers are always reported regardless of the value or presence of this option.

-ignored_cells

Reports all the ignored cells during physical banking along with the reason.

DESCRIPTION

This command lists information about multibit registers and multi- voltage cells in the design. The report includes the total number of single-bit and multibit sequential cells, the single-bit equivalent of these sequential cells, the sequential cell and flip-flop banking ratio, and the multibit register decomposition. If multi-voltage cells have been included using the -mv_cell option, the report will also

contain the total number of single-bit and multibit cells, the banking ratio, and the multibit cell decomposition for the specified multi-voltage cells.

The sequential cells only include leaf cells. Clock-gating latches are excluded in this report. Therefore, the sequential cell count reported here can differ from what is reported by the report_qor and report_area commands.

The multibit cell decomposition section reports the information about multibit cell references used in the design. See the example section for more details.

When you specify the -hierarchical option, the number of single-bit and multibit cells and the banking ratio are reported for each subdesign.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example generates a multibit banking report

```
prompt> report_multibit

****************************************
Report : report_multibit
Design : create_placement
Version: O-2018.06-SP2-BETA
Date   : Fri Jul 20 01:10:38 2018

****************************************

Total number of sequential cells:                4020
  Number of single-bit flip-flops:               3149
  Number of single-bit latches:                  29
  Number of multi-bit flip-flops:                842
  Number of multi-bit latches:                   0

Total number of single-bit equivalent cells:     8530
  (A)  Single-bit flip-flops:                    3149
  (B)  Single-bit latches:                       29
  (C)  Multi-bit flip-flops:                     5352
  (D)  Multi-bit latches:                        0

Sequential cells banking ratio ((C+D)/(A+B+C+D)):    62.74%
Flip-flop cells banking ratio ((C)/(A+C)):           62.96%
BitsPerflop:                                         2.13

Multi-bit Register Decomposition:
--------------------------------------------------------------------------------
Bit-Width  Reference              Number of instances    Single-bit Equivalent
--------------------------------------------------------------------------------
2 -bit                      124 ( 3.08% )        248 ( 2.91% )

        CMM1FD2EQB2V2H1X1       124
```

```
    4 -bit                  160 ( 3.98% )       640 ( 7.50% )

          CMM1FD2EQB4V4H1X1      160

    8 -bit                  558 ( 13.88% )      4464 ( 52.33% )

          CMM1FD2EQB8V8H1X1      558


    ---------------------------------------------------------------------------

    1
```

SEE ALSO

create_multibit(2)
identify_multibit(2)

report_multisource_clock_sink_groups

Report sink groups defined for multisource clock tap assignment

SYNTAX

status **report_multisource_clock_sink_groups**
[*sink_groups*]

Data Types

sink_groups collection

ARGUMENTS

sink_groups

Specifies a list of sink groups to be reported. Sink groups are defined using the **create_multisource_clock_sink_group** command. This option is optional. By default, the command will report all defined sink groups.

DESCRIPTION

The **report_multisource_clock_sink_group** command is used to report sink groups defined for multisource clock tap assignment. It will print the name, the type, the driver object as well as the list of sinks of a sink group.

The argument **sink_groups** can be used to restrict reporting to the given list of sink groups.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This example reports all defined sink groups.

```
report_multisource_clock_sink_groups
```

This example reports the details of sink group sink_group1.

```
report_multisource_clock_sink_groups sink_group1
```

This example reports a collection of sink groups.

```
report_multisource_clock_sink_groups [get_multisource_clock_Sink_groups sink_group*]
```

SEE ALSO

create_multisource_clock_sink_group(2)
remove_multisource_clock_sink_groups(2)
get_multisource_clock_sink_groups(2)
add_to_multisource_clock_sink_group(2)
remove_from_multisource_clock_sink_group(2)
synthesize_multisource_clock_taps(2)
set_multisource_clock_tap_options(2)

report_multisource_clock_subtree_constraints

Reports constraints on cell and pins for structural synthesis of multisource clock tree synthesis (CTS) subtrees.

SYNTAX

status **report_multisource_clock_subtree_constraints**
[-names *name_list*]
[-clocks *clock_list*]

Data Types

name_list list of names
clock_list collection of clocks

ARGUMENTS

-names *name_list*

Specifies the subtree sets for which to report constraints on pin or cells. If neither this option nor the **-clocks** option are specified, all constraints are reported for all subtree option sets.

-clocks *clock_list*

Specifies the clocks for which to report constraints on pin or cells. If neither this option nor the **-names** option are specified, then all constraints are reported for all subtree option sets.

DESCRIPTION

The **report_multisource_clock_subtree_constraints** command reports constraints on cells or pins. Use this command in addition to the **report_multisource_clock_subtree_options** command for reporting cell- or pin-specific options or constraints.

Use the **set_multisource_clock_subtree_constraints** command to set constraints which can be cleared with this command. Use the **remove_multisource_clock_subtree_constraints** command to remove constraints. Use the **synthesize_multisource_clock_subtrees** command to synthesize the subtrees, as per the constraints.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This example reports all constraints:

prompt> **report_multisource_clock_subtree_constraints**

This example reports clock-specific constraints:

prompt> **report_multisource_clock_subtree_constraints -clocks [get_clock clk]**

This example reports name-specific constraints:

prompt> **report_multisource_clock_subtree_constraints -names clk**

SEE ALSO

remove_multisource_clock_subtree_constraints(2)
remove_multisource_clock_subtree_options(2)
report_multisource_clock_subtree_options(2)
set_multisource_clock_subtree_constraints(2)
set_multisource_clock_subtree_options(2)
synthesize_multisource_clock_subtrees(2)

report_multisource_clock_subtree_options

Reports the settings defined by set_multisource_clock_subtree_options.

SYNTAX

string **report_multisource_clock_subtree_options**
[-names *string_list*]
[-clocks *clock_list*]

Data Types

string_list list
clock_list collection

ARGUMENTS

-names *string_list*

Restricts the command to report the subtree options only for the specified subtree sets. By default, all sets are reported.

-clocks *clock_list*

Restricts the command to report the subtree sets only for the specified clocks. By default, all subtree sets are reported.

DESCRIPTION

The **report_multisource_clock_subtree_options** command is used to report the subtree synthesis constraints applied by the **set_multisource_clock_subtree_options** command.

This report includes the following information for every subtree set: The name, the clock to be balanced in that set, the driver objects of that set, and any *max_total_wire_delay* constraint applied for the set, along with the corner or corners for which that *max_total_wire_delay* constraint applies.

By default, this command reports the constraints for all defined subtree sets. If the -names option is used, then only the specified sets will be reported. If the -clocks option is used, then all subtree sets defined on the given clocks are reported. You can apply -names together with -clocks.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This example reports the constraints for all subtree sets:

```
report_multisource_clock_subtree_options
```

This example reports the constraints for only subtree sets *clock_set1* and *clock_set2*:

```
report_multisource_clock_subtree_options -names "clock_set1 clock_set2"
```

This example reports the constraints only for clock *clk*:

```
report_multisource_clock_subtree_options -clocks [get_clock clk]
```

SEE ALSO

set_multisource_clock_subtree_options(2)
remove_multisource_clock_subtree_options(2)
synthesize_multisource_clock_subtrees(2)

report_multisource_clock_tap_options

Reports the settings defined by **set_multisource_clock_tap_options**.

SYNTAX

string **report_multisource_clock_tap_options**
[-names *string_list*]
[-clocks *clock_list*]

Data Types

string_list list
clock_list collection

ARGUMENTS

-names *string_list*

Restricts the command to report the tap assignment options only for the specified names. The default name of an option set is the name of the clock of that set. By default, all option sets are reported.

-clocks *clock_list*

Restricts the command to report the options only for the specified clocks. By default, all option sets are reported.

DESCRIPTION

The **report_multisource_clock_tap_options** command is used to report the tap assignment settings applied by the **set_multisource_clock_tap_options** command.

This report includes the following information for every option set: The name, the clock, the tap driver objects, and the number of tap drivers of that group.

By default, this command reports the constraints for all defined option sets. If the **-names** option is used, then only the specified option sets will be reported. If the **-clocks** option is used, then all tap assignment option sets defined on the given clocks are reported. You can apply **-names** together with **-clocks**. In this case all option sets with the given names or that have one of the specified clocks are reported.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This example reports the constraints for all tap assignment options:

```
prompt> report_multisource_clock_tap_options
```

This example reports the constraints for only option sets clock_set1 and clock_set2:

```
prompt> report_multisource_clock_tap_options -names "clock_set1 \
  clock_set2"
```

This example reports the constraints only for clock *clk*:

```
prompt> report_multisource_clock_tap_options -clocks [get_clock clk]
```

SEE ALSO

remove_multisource_clock_tap_options(2)
set_multisource_clock_tap_options(2)
synthesize_multisource_clock_taps(2)

report_multistage_timing

Reports timing information for stages preceding and following the specified timing endpoints.

SYNTAX

status **report_multistage_timing**
[-scenarios scenario_list]
[-groups group_list]
[-through endpoint_list]
[-max_paths max_path_count]
[-max_stages max_stage_count]
[-pba_mode none | path | exhaustive]
[-significant_digits digits]
[-physical]
[-verbose]

scenario_list list *group_list* list *endpoint_list* list *max_path_count* integer *max_stage_count* integer

ARGUMENTS

-scenarios scenario_list

Specifies the scenarios for which to generate the timing report. Each of the specified scenarios is evaluated by the report. If the **-scenarios** option is not specified, the report considers every active scenario of the design.

-groups group_list

Specify the list of path group names for reporting paths. Default is all path groups.

-max_paths max_path_count

Specifies the number of paths to report. The default value is 1.

-max_stages max_stage_count

Specifies the number of stages to report before and after the specified stage endpoints. The default is 3, meaning that the report includes 3+1 stages before and 3 stages after the specified timing endpoints.

-pba_mode none | path | exhaustive

Controls path-based analysis. See the report_timing command man page for more details.

-significant_digits digits

Specifies the number of significant digits (digits to the right of the decimal point) to report. Allowed values are from 0 through 13. Using this option overrides the value set by the **shell.common.report_default_significant_digits** application option.

-physical

Adds cell location information to the report. The **Manhattan Distance** column reports the half-perimeter distance between the startpoint and endpoint of the reported timing path. The report also reports the summation of the half-perimeter distance between all ports and cells on the datapath of the reported timing path.

-verbose

Additional information will be displayed, causing the report to become wider.

DESCRIPTION

The report_multistage_timing command generates a report that contains timing information for the current design. By default, the report_multistage_timing command reports for the single worst slack path in the design.

Command options let you specify the number of paths to report and the number of stages to report before and after the specified stages. Since the timing endpoint of a stage is also the timing startpoint of the subsequent stage, an additional stage is reported before the requested number of stages, giving the start point of the first requested stage. If a timing loop is detected, the report will be truncated at that stage.

Column Descriptions

Stage : The number of stages either preceding (negative value) or following (postive value) the specified stage.
Path Delay: The maximum delay value of the timing path.
Repeaters : A count of the number of repeater cells (buffers or inverters) in the timing path.
EP Latency: The capture clock latency to the timing endpoint (the launch clock latency would be shown as the EP Latency in the pri
CRP : Common path pessimism between the launch and capture clock paths.
Skew : Difference between the capture clock latency and the launch clock latency.
Slack : Worst timing slack at the timing endpoint
Levels : Number of two-input equivalent gates in the timing path.
Scenario : Timing Scenario of the timing path.

Multicorner-Multimode Support

By default, this command considers information from all active scenarios.

EXAMPLES

The following example reports multistage timing information for three stages ahead of and beyond the stage to the worst timing endpoint in the design.

```
prompt> report_multistage_timing -significant_digits 6 -verbose
****************************************
Report : multistage_timing
    -max_paths 1
    -max_stages 3
    -scenarios func::default
Design : leon3mp
Version: O-2018.06-SP5
Date   : Tue Mar 05 12:53:30 2019
```

```
****************************************

 Stage Endpoint                                                                Path Delay Repeaters EP Latency CRP     Skew     Slack     Levels Scenario
 ----- ------------------------------------------------------------------------ ---------- --------- ---------- -------- --------- --------- ------ -------------
   -4 u0_1/p0/iu0/r_reg_D__SET__1_/D (SDFFX1_LVT)                    3.553104         5   3.500000 0.000000 -0.500000  0.457022
   -3 u0_1/p0/iu0/r_reg_D__SET__1_/D (SDFFX1_LVT)                    3.553104         5   3.500000 0.000000 -0.500000  0.457022
   -2 u0_1/p0/c0mmu/icache0_r_reg_REQ_/D (SDFFSSRX1_LVT)             3.536396         5   3.500000 0.000000 -0.500000  0.34
   -1 u_misc_new/u_misc_new/u_m_ahb0/r_reg_HSLAVE__2_/D (SDFFSSRX1_LVT)  3.272438     5   3.500000 0.000000 -0.5
    0 u0_3/p0/c0mmu/m0/itlb0_tag0_4_r_reg_BTAG__PPN__10_/D (SDFFX1_LVT)  4.391747     4   0.036011 0.000000 -3.963
    1 u0_3/p0/c0mmu/icache0_r_reg_WADDRESS__22_/D (SDFFX1_LVT)          3.093261     0   3.500000 0.000000  3.462978
    2 u_misc_new/u_misc_new/u_m_sdc/r_reg_ADDRESS__15_/D (SDFFX1_LVT)  2.490597     2   3.500000 0.000000 -0.500
    3 u_misc_new/u_misc_new/u_m_sdc/r_reg_ADDRESS__15_/D (SDFFX1_LVT)  0.249786     0   3.500000 0.000000 -0.500

1
```

SEE ALSO

report_timing(2)
shell.common.report_default_significant_digits(3)

report_mv_cells

Report information related to multi-voltage cells in the design.

SYNTAX

status **report_mv_cells**
[*cell_list*]
[-all]
[-enable_level_shifter]
[-level_shifter]
[-isolation]
[-repeater]
[-retention]
[-retention_clamp]
[-always_on]
[-verbose]
[-output <*output_file*>]
[-html]
[-csv]
[-nosplit]
[-power_domain <*power_domain_list*>]
[-power_strategy <*power_strategy_list*>]

Data Types

cell_list list
output_file string
power_domain_list list
power_strategy_list list

ARGUMENT

cell_list

Specify a list of library cells to report.

-all

Report enable level shifter, level shifter, isolation, repeater, retention register, retention clamp, always-on buffer/inverter/tie, and power switch cells. This is the default setting.

-enable_level_shifter

Report enable level shifter cells only.

-level_shifter

Report level shifter cells only.

-isolation

Report isolation cells only.

-repeater

Report repeater cells only.

-retention

Report retention register cells only.

-retention_clamp

Report retention clamp cells only.

-always_on

Report always-on buffers, inverters, and tie lib cells only.

-switch

Report power switch lib cells only.

-verbose

Report detailed information about a cell's attributes.

-output <*output_file*>

Print report into <*output_file*>.

-html

Print report in HTML format.

-csv

Print report in CSV (Comma-Separated Value) format. *-output* option is required when using this option and cannot be used in conjuntion with the *-html* option

-nosplit

Most of the reported data is listed in fixed-width columns. If the data for a given column exceeds the column width, the next field begins on a new line, starting in the right column. The *-nosplit* option prevents this behavior, allowing all the data to be written on a single line.

-power_domain *<power_domain_list>*

Specify a list of power domain elements to use for filtering. When issued, only cells that reside inside any of the power domains listed in *<power_domain_list>* will be reported.

-power_strategy *<power_strategy_list>*

Specify a list of power strategy elements to use for filtering. When issued, only cells that are associated to any of the strategies listed in *<power_strategy_list>* will be reported. This filter only applies to cell instances that can be associated to a power strategy

DESCRIPTION

This command reports multi-voltage related cell information. It includes power management attributes defined on special cells, such as level shifter, enable level shifter, isolation, repeater, retention, retention clamp, and always on cells. In verbose mode, more detailed information about about the cells will be provided.

EXAMPLES

The following example reports all the repeater cells in the design:

```
prompt> report_mv_cells -repeater
****************************************
Report : report_mv_cells
       -repeater
Design : top
 ...
****************************************

Signal Pin Attributes
    lse - level shifter enable pin
   isoe - isolation cell enable pin
   scmr - standard cell main rail
    sav - retention save pin
    rst - retention restore pin

Supply Pin Attributes
    PRM - primary
    BCK - backup
    INT - internal
    PWL - P-well
    NWL - N-well
    DPL - deep P-well
    DNL - deep N-well

Cell Attributes
     dt - dont touch
     ao - always-on
     ls - level shifter
    els - enable level shifter
    iso - isolation cell
    ret - retention cell
    zpr - zero pin retention cell
     sw - switch cell
     cg - coarse grain
     fg - fine grain
    mac - macro cell
     io - IO pad cell
    hvt - in high threshold voltage group
    nvt - in normal threshold voltage group
    lvt - in low threshold voltage group

--------------------
```

```
 Repeater Cells (3)
---------------------

 Cell Name          Location Domain Strategy (Domain)  Lib Cell
 ---------------------- ---------------- ------------------- ---------------------
 mid1/snps_MID__RPTR_snps_in1_UPF_RPTR
                MID          RPTR(MID)        CORE_NPPL_umc13ell/BUFX10MTH
 mid1/snps_MID__RPTR_snps_in2_UPF_RPTR
                MID          RPTR(MID)        CORE_NPPL_umc13ell/BUFX10MTH
 mid1/snps_MID__RPTR_snps_in3_UPF_RPTR
                MID          RPTR(MID)        CORE_NPPL_umc13ell/BUFX10MTH

1
```

SEE ALSO

report_cell(2)
report_mv_lib_cells(2)

report_mv_design

Reports a summary of power management cells.

SYNTAX

status **report_mv_design**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command will print a summary of power management cells in the design. The summary includes the following:

- Number of power domains
- Number of supply nets/sets
- Number and details of voltage areas
- Number of power management cells inserted for each type
- Status of UPF commit

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example prints a summary of power management cells in the design:

prompt> **report_mv_design**

SEE ALSO

check_mv_design(2)
compile(2)

report_mv_lib_cells

Report information related to multi-voltage library cells in the design.

SYNTAX

status **report_mv_lib_cells**
[*lib_cell_list*]
[-all]
[-enable_level_shifter]
[-level_shifter]
[-isolation]
[-retention]
[-always_on]
[-switch]
[-macro]
[-io_pad]
[-standard]
[-diode]
[-buffer]
[-output *output_file*]
[-html]
[-csv]
[-example]
[-verbose]

ARGUMENT

libcell_list

Specifies a list of library cells to report.

-all

Report enable level shifter, level shifter, isolation, retention register, always-on buffer/inverter/tie, and power switch lib cells. This is the default setting.

-enable_level_shifter

Report enable level shifter lib cells only.

-level_shifter

Report level shifter lib cells only.

-isolation

Report isolation lib cells only.

-retention

Report retention register lib cells only.

-always_on

Report always-on buffers, inverters, and tie lib cells only.

-switch

Report power switch only.

-macro

Report macro lib cells only.

-io_pad

Report IO pad lib cells only.

-standard

Report Standard lib cells only. Standard cells are not included in -all option.

-diode

Report diode lib cells only.

-buffer

Report single-rail and dual-rail buffer and inverter lib cells only. Single-rail buffer and inverter lib cells are not included in -all option.

-verbose

Report detailed information about lib cell attributes and pins.

-output <*output_file*>

Print report into <*output_file*>.

-html

Print report in HTML format.

-csv

Print report in CSV (Comma-Separated Value) format.

-example

Display a list of typical example usages of this command.

-verbose

Displays all attributes and pins of each library cell.

DESCRIPTION

This command reports multivoltage-related library cell information. It includes power management attributes defined on special cells, such as level shifter, enable level shifter, isolation, retention, switch, always-on buffer, always-on inverter, always-on tie, macro, I/O pad, and standard library cells.

In verbose mode, it also displays detailed information about the data, power, and ground pins.

EXAMPLES

The following example reports all the retention cells in the design:

```
prompt> report_mv_lib_cells -retention
****************************************
Report : report_mv_lib_cells
       -retention
Design : simple
Version: P-2019.03-DEV
Date   : Fri May  4 14:06:49 2018
****************************************

Signal Pin Attributes
     isoe - isolation cell enable pin
      lse - level shifter enable pin
      rst - retention restore pin
      sav - retention save pin

Supply Pin Attributes
     bkup - backup
   dnwell - deep N-well
   dpwell - deep P-well
      int - internal
    nwell - N-well
     prim - primary
    pwell - P-well
     scmr - standard cell main rail

Cell Attributes
       ao - always-on
       cg - coarse grain
     diff - differential LS
       dt - dont touch
      els - enable level shifter
       fg - fine grain
      hvt - high threshold voltage group
       io - IO pad cell
      iso - isolation cell
    isols - ISO-LS style ELS
       ls - level shifter
```

```
      lvt - low threshold voltage group
      mac - macro cell
      nor - NOR isolation or ELS
      nvt - normal threshold voltage group
       od - overdriven LS
      ret - retention cell
       sw - switch cell
      zpr - zero pin retention cell

----------------------
 Retention Cells (1)
----------------------

ts16ncgllogl16hdp090f_frame_timing_all|ts16ncgllogl16hdp090f_frame_timing_ccs/HDPLVT16_FSB2DPQ_PT_1

  Attributes: ret, bias, nvt, purpose=all, Q=n/a
 Signal Pins: B1(VDDR/VSS/VBP/VBN/sav), B2(VDDR/VSS/VBP/VBN/rst),
   D(VDD/VSS/VBP/VBN), SI(VDD/VSS/VBP/VBN), SE(VDD/VSS/VBP/VBN),
   CK(VDD/VSS/VBP/VBN), Q(VDD/VSS/VBP/VBN)
 Supply Pins: VDD(pwr/prim), VDDR(pwr/bkup), VBN(pwell), VBP(nwell),
   VSS(gnd/prim)

 Pane  VBN    VBP    VDD    VDDR   VSS    Temp  Proc  Label
 ----- ------ ------ ------ ------ ------ ----- ----- -----
    0  0.000  0.720  0.720  0.720  0.000   -40  1.000   -
    1  0.000  0.800  0.800  0.800  0.000    85  1.000   -
    2  0.000  0.880  0.880  0.880  0.000   125  1.000   -
    3  0.000  0.630  0.630  0.630  0.000   -40  1.000   -
    4  0.000  0.700  0.700  0.700  0.000    85  1.000   -
    5  0.000  0.770  0.770  0.770  0.000   125  1.000   -
    6  0.000  0.810  0.810  0.810  0.000   -40  1.000   -
    7  0.000  0.900  0.900  0.900  0.000    85  1.000   -
    8  0.000  0.990  0.990  0.990  0.000   125  1.000   -

--------------------------------------------------------------------------------
```

SEE ALSO

report_lib(2)
report_lib_cells(2)
report_lib_pins(2)

report_mv_path

Reports multivoltage paths with multivoltage constraints and multivoltage cell insertion and association situations.

SYNTAX

status **report_mv_path**
[-isolation]
[-level_shifter]
[-repeater]
[-all_path]
[-pin *pin*]
[-net *net*]
[-shifting]
[-full_path]
[-applied_strategy_only]
[-cell *cell*]
[-all_not_associated]
[-all_heterogeneous_paths]
[-name]

Data Types

pin string
net string
cell string

ARGUMENTS

-isolation

Reports the path or isolation cell with isolation constraints. This is the default option.

-level_shifter

Reports the path or level shifter cell with level shifter constraints.

-repeater

Reports the path or repeater cell with repeater constraints.

-all_path

Reports all paths in the design.

-pin *pin*

Reports the path which the pin is included.

-net ***net***

Reports the path which the net segment is included.

-shifting

Reports the path voltage shifting situation with level shifter constraints.

-full_path

Reports full isolation path information, including skipped buffers and inverters in the path. You must also specify the **-isolation** option.

-applied_strategy_only

Reports applied strategy on boundary port only

-cell ***cell***

Reports cell isolation or level shifter related information for the specified *cell*.

-all_not_associated

Reports all unassociated isolation or level shifter cells in the design.

-all_heterogeneous_paths

Report all heterogeneous path only with multiple supplies details, homogeneous paths are not reported. This option cannot be combined with any other option, all other options will be ignored. The report will only include all heterogeneous path information.

-name

Reports name based match information on isolation path.

DESCRIPTION

This command reports the path with multivoltage constraints and all details related to multivoltage cell insertion and association. This command helps you analyze the path situation if there is isolation and voltage shifting violation reported by **check_mv_design**.

The command reports either isolation path, level shifter path or repeater path constraints. You can report situations where the driving and loading conditions are different between isolation, level shifter or repeater paths. Only one type of constraints is reported, the default is **-isolation**. To report all types, run the command multiple times with different options.

EXAMPLES

The following example reports a level shifter cell in the design which is associated with UPF strategy.

```
prompt> report_mv_path -cell h1/CLK_UPF_LS
****************************************
Analysis : MV Path
 ...
```

```
****************************************
MV Cells Condition :

----------------------------------------
Cell Name              : h1/CLK_UPF_LS
Cell Type              : Level Shifter Cell
Power Domain           : h1/PD
Library Cell           : LVLHLX8M
Associated Strategy    : PD_LS_strategy_1(h1/PD)

Power Supplies             -- Input --              -- Output --
                     VDD90(Domain Default)      VDD90(Domain Default)

 Input Driver          : CLK
   Power Supply        : VDD108
 Output Loads          : h1/reg_2_/CK; h1/reg_1_/CK;
   Power Supply        : h1/VDD90
 Output Loads          : h1/h2/reg_2_/CK; h1/h2/reg_1_/CK;
   Power Supply        : h1/h2/VDD90
----------------------------------------
1
```

The following example reports a level shifter cell which cannot be associated with any UPF strategy. The reason for failure is also listed.

```
prompt> commit_upf
Information: Power intent has been successfully committed. (UPF-072)
Information: Total 0 isolation cell in the design. (MV-021)
Information: Total 5 out of 6 level shifter cells have been associated with strategy. (MV-081)
Information: Total 0 enable level shifter cell in the design. (MV-021)
Information: Total 0 retention cell in the design. (MV-021)
Error: Power cell h1/CLK_UPF_LS is not associated
 with any power strategies or supply net connections. (UPF-097)
Error: problem in commit_upf
       Use error_info for more info. (CMD-013)
```

The **commit_upf** command reports a UPF-097 failure during level shifter cell association. You can use the **report_mv_path** command to check the path and cell MV constraints and failure reason:

```
prompt> report_mv_path -cell h1/CLK_UPF_LS
****************************************
Analysis : MV Path
 ...

****************************************
MV Cells Condition :

----------------------------------------
Cell Name              : h1/CLK_UPF_LS
Cell Type              : Level Shifter Cell
Power Domain           : h1/PD
Library Cell           : LVLHLX8M
Associated Strategy    : None
Association Error      :
                     PD_LS_strategy_1(h1/PD) --
  Level Shifter instance output loads have multiple different voltage supply nets
                     PD_LS_strategy_2(h1/h2/PD) --
```

```
 Level Shifter instance output loads have multiple different voltage supply nets

Power Supplies              -- Input --              -- Output --
                    VDD90(Domain Default)       VDD90(Domain Default)

 Input Driver          : CLK
   Power Supply        : VDD108
 Output Loads          : h1/h2/reg_2_/CK;
   Power Supply        : VDD108
 Output Loads          : h1/reg_2_/CK; h1/reg_1_/CK;
   Power Supply        : h1/VDD90
 Output Loads          : h1/h2/reg_1_/CK;
   Power Supply        : h1/h2/VDD90
```

The following example reports an isolation shifter cell in the design which is associated with UPF strategy.

```
prompt> report_mv_path -cell UPC_BLK/CIN_UPF_ISO
****************************************
Analysis : MV Path
 ...

****************************************
MV Cells Condition :

----------------------------------------
Cell Name             : UPC_BLK/CIN_UPF_ISO
Cell Type             : Isolation Cell
Power Domain          : pd5
Library Cell          : ISOLNX2M
Associated Strategy   : iso_stra_5(pd5)

Power Supplies              -- Input --              -- Output --
                    VDD(Domain Default)         VDD(Domain Default)

 Input Driver          : CIN
   Power Supply        : VDDST
 Output Loads          : UPC_BLK/U46/B; UPC_BLK/U30/A1;
   Power Supply        : VDD
----------------------------------------
1
```

The following example reports an isolation shifter cell in the design which cannot be associated with any isolation strategy with the reason of failure.

```
prompt> report_mv_path -cell UPC_BLK/CIN_UPF_ISO
****************************************
Analysis : MV Path
 ...

****************************************
MV constraints and path condition :

----------------------------------------
UPC_BLK/CIN_UPF_ISO
Cell Type          : Isolation Cell
Library Cell       : ISOLNX2M
Associated Strategy : None
```

```
Association Error      : iso_stra_5 --
  Isolation instance clamp value doesn't match with strategy

Power Supplies          -- Input --             -- Output --
                     VDD(Domain)             VDD(Domain)

  Input Driver        : iso_cntl_2
    Power Supply      : VDD
  Output Loads        : UPC_BLK/U30/A1; UPC_BLK/U46/B;
    Power Supply      : VDD
----------------------------------------
1
```

The following example reports a path voltage shifting conditions. The power domains and strategies of this design are shown in the first example:

```
prompt> report_power_domain [get_power_domains -hierarchical]
****************************************
Report : Power Domain
Corner : sc1.BC.corner
 ...

****************************************
--------------------------------------------------------------------------------
Power Domain          : TOP
Current Scope         : top (top level)
Elements              : top, h1/h2/h3
Voltage Area          : DEFAULT_VA
Available Supply Nets  : VDD108, VDD90, VSS
Available Supply Sets  :

Default Supplies         -- Power --             -- Ground --
  Primary             : VDD108 [1.08]           VSS [0.00]
  Isolation           : --                   --
  Retention           : --                    --

Power Switch          :
Isolation Strategy    :
Level Shifter Strategy  : TOP_LS_strategy_1
  Applies To          : both
  Rule                : both
  Location            : automatic
  Elements            : N/A
  Mapped Cells        : N/A

Retention Strategy    :
--------------------------------------------------------------------------------
Power Domain          : h1/PD
Current Scope         : top (top level)
Elements              : h1
Voltage Area          : h1/PD
Available Supply Nets  : h1/VDD90, h1/VDD108, h1/VSS
Available Supply Sets  :

Default Supplies         -- Power --             -- Ground --
  Primary             : h1/VDD90 [0.90]         h1/VSS [0.00]
  Isolation           : --                   --
```

```
  Retention          : --                    --

Power Switch         :
Isolation Strategy   :
Level Shifter Strategy : PD_LS_strategy_1
  Applies To         : both
  Rule               : both
  Location           : automatic
  Elements           : N/A
  Mapped Cells       : N/A

Retention Strategy   :
--------------------------------------------------------------------------------
Power Domain         : h1/h2/PD
Current Scope        : top (top level)
Elements             : h1/h2
Voltage Area         : h1/h2/PD
Available Supply Nets  : h1/h2/VDD90, h1/h2/VDD108, h1/h2/VSS
Available Supply Sets  :

Default Supplies         -- Power --              -- Ground --
  Primary            : h1/h2/VDD90 [0.90]       h1/h2/VSS [0.00]
  Isolation          : --                    --
  Retention          : --                    --

Power Switch         :
Isolation Strategy   :
Level Shifter Strategy : PD_LS_strategy_2
  Applies To         : both
  Rule               : both
  Location           : automatic
  Elements           : N/A
  Mapped Cells       : N/A

Retention Strategy   :
--------------------------------------------------------------------------------
1

prompt> report_mv_path -pin h1/h2/reg_2_/CK -level_shifter -shifting
****************************************
Analysis : MV Path
 ...

****************************************
MV Level Shifter Constraints And Path Condition :

----------------------------------------
Path with CLK :

Level0--CLK
  Pin/Port Type        : Driver Port
  Supplies             : VDD108(Related)/VSS(Domain)
  PST Voltage          : (1.08, 1.08)/(0, 0)
  Applied Strategy     : TOP_LS_strategy_1(TOP)
   Rule                : Both
   Applies To          : Both
```

```
    Threshold            : N/A(Default 0.0)
    Location             : Automatic
    Hi/Low               : HiConn
    Location Self Domain   : TOP
    Location Parent Domain  : N/A
   Multi Load Supplies     : Yes

  Level1--h1/CLK
    Pin/Port Type        : Domain Boundary
    Applied Strategy     : PD_LS_strategy_1(h1/PD)
     Rule                : Both
     Applies To          : Both
     Threshold           : N/A(Default 0.0)
     Location            : Automatic
     Hi/Low              : HiConn
     Location Self Domain   : h1/PD
     Location Parent Domain  : TOP
    Applied Strategy     : TOP_LS_strategy_1(TOP)
     Rule                : Both
     Applies To          : Both
     Threshold           : N/A(Default 0.0)
     Location            : Automatic
     Hi/Low              : LowConn
     Location Self Domain   : h1/PD
     Location Parent Domain  : TOP
    Multi Load Supplies     : Yes

  Level2--h1/CLK_UPF_LS/Y
    Pin/Port Type        : Path Pin

  Cell Name              : h1/CLK_UPF_LS
  Cell Type              : Level Shifter Cell
  Power Domain           : h1/PD
  Library Cell           : LVLHLX8M
  Associated Strategy    : None
  Association Error      :
                    PD_LS_strategy_1(h1/PD) --
    Level Shifter instance output loads have multiple different voltage supply nets
                    PD_LS_strategy_2(h1/h2/PD) --
    Level Shifter instance output loads have multiple different voltage supply nets

  Power Supplies            -- Input --              -- Output --
                    h1/VDD90(Domain Default)    h1/VDD90(Domain Default)

  Level3--h1/reg_1_/CK
    Pin/Port Type        : Load Pin
    Supplies             : VDD90(Domain)/VSS(Domain)
    PST Voltage          : (0.9, 0.9)/(0, 0)
  Shifting               : High To Low

  Level3--h1/reg_2_/CK
    Pin/Port Type        : Load Pin
    Supplies             : VDD90(Domain)/VSS(Domain)
    PST Voltage          : (0.9, 0.9)/(0, 0)
  Shifting               : High To Low
```

```
Level3--h1/h2/CLK
  Pin/Port Type          : Domain Boundary
  Applied Strategy       : PD_LS_strategy_2(h1/h2/PD)
   Rule                  : Both
   Applies To            : Both
   Threshold             : N/A(Default 0.0)
   Location              : Automatic
   Hi/Low                : HiConn
   Location Self Domain  : h1/h2/PD
   Location Parent Domain : h1/PD
  Applied Strategy       : PD_LS_strategy_1(h1/PD)
   Rule                  : Both
   Applies To            : Both
   Threshold             : N/A(Default 0.0)
   Location              : Automatic
   Hi/Low                : LowConn
   Location Self Domain  : h1/h2/PD
   Location Parent Domain : h1/PD
  Multi Load Supplies    : Yes

Level4--h1/h2/reg_1_/CK
  Pin/Port Type          : Load Pin
  Supplies               : VDD90(Domain)/VSS(Domain)
  PST Voltage            : (0.9, 0.9)/(0, 0)
Shifting                 : High To Low

Level4--h1/h2/reg_2_/CK
  Pin/Port Type          : Load Pin
  Supplies               : VDD108(Related)/VSS(Domain)
  PST Voltage            : (1.08, 1.08)/(0, 0)
Shifting                 : No Shifting

Level4--h1/h2/h3/CLK
  Pin/Port Type          : Domain Boundary
  Applied Strategy       : TOP_LS_strategy_1(TOP)
   Rule                  : Both
   Applies To            : Both
   Threshold             : N/A(Default 0.0)
   Location              : Automatic
   Hi/Low                : HiConn
   Location Self Domain  : TOP
   Location Parent Domain : h1/h2/PD
  Applied Strategy       : PD_LS_strategy_2(h1/h2/PD)
   Rule                  : Both
   Applies To            : Both
   Threshold             : N/A(Default 0.0)
   Location              : Automatic
   Hi/Low                : LowConn
   Location Self Domain  : TOP
   Location Parent Domain : h1/h2/PD
  Multi Load Supplies    : Yes

----------------------------------------
1
```

The following example skips buffers and inverters along the isolation path. By default, **report_mv_path** will not report those buffers

and inverters that are skipped. The **-full_path** option includes all buffers and inverters in the isolation path.

The first example does not use **-full_path**, the second example uses **-full_path**.

```
prompt> report_mv_path -pin u_des_ctl/U42/Z -isolation
****************************************
Analysis : MV Path
 ...

****************************************
MV Isolation Constraints And Path Condition :

----------------------------------------
Path with u_des_ctl/U42/Z :

Level0--u_des_ctl/U42/Z
  Pin/Port Type          : Driver Pin
  Supplies               : VDD_SD_1(Domain)/VSS(Domain)
  Homogeneous            : No
  Diff Only              : No

Level1--u_des_ctl/macro_reset_n
<details>

Level2--u_des_hard_macro/reset_n
<details>

Level4--u_des_hard_macro/u_des_hard_macro_bypass/reset_n
<details>

Level5--u_des_hard_macro/u_des_hard_macro_bypass/.../Z
<details>

Level6--u_des_hard_macro/u_des_hard_macro_bypass/o_tval_reg/CDN
<details>

prompt> report_mv_path -pin u_des_ctl/U42/Z -isolation -full_path
****************************************
Analysis : MV Path
 ...

****************************************
MV Isolation Constraints And Path Condition :

----------------------------------------
Path with u_des_ctl/U42/Z :

Level0--u_des_ctl/U42/Z
  Pin/Port Type          : Driver Pin
  Supplies               : VDD_SD_1(Domain)/VSS(Domain)
  Homogeneous            : No
  Diff Only              : No

Level1--u_des_ctl/macro_reset_n
<details>

Level2--u_des_hard_macro/reset_n
```

<details>

Level3--u_des_hard_macro/U3/Z
Pin/Port Type : Path Pin

Level4--u_des_hard_macro/u_des_hard_macro_bypass/reset_n
<details>

Level5--u_des_hard_macro/u_des_hard_macro_bypass/.../Z
<details>

Level6--u_des_hard_macro/u_des_hard_macro_bypass/o_tval_reg/CDN
<details>

prompt> **report_mv_path -all_heterogeneous_paths** Information: report all heterogeneous paths only with -all_heterogeneous_paths option, any other option will be ignored. (MV-612) ** Analysis : MV Path Design : CORE Version : Q-2019.12-SP2-DEV Date : Fri Jan 10 10:30:52 2020 ** MV Path Heterogeneous Fanouts Condition :

---------- Path with D[11] :

Level 0 -- D[11] Pin/Port Type : Driver Port Supplies : VDDST(Domain),VSS(Domain) Multi Load Supplies : Yes Load Supplies : (VDD,VSS) (VDDL,VSS)

Level 1 -- MUX_OUT_BLK/DATA[11] Pin/Port Type : Domain Boundary Single Load Supply : Yes Load Supply : (VDD,VSS)

Level 2 -- MUX_OUT_BLK/DATA[11]_UPF_ISO_snps_pd5__iso_stra_5_snps_DATA[11]_UPF_ISO/A Pin/Port Type : Load Pin Supplies : VDD(Domain),VSS(Domain) Cell Name : MUX_OUT_BLK/DATA[11]_UPF_ISO_snps_pd5__iso_stra_5_snps_DATA[11]_UPF_ISO Cell Type : Isolation Cell Power Domain : pd5 Library Cell : ISOLNX2M Associated Strategy : iso_stra_5(pd5)

Power Supplies -- Input -- -- Output -- VDD(Domain Default) VDD(Domain Default)

Level 1 -- REG_BLK/DATA_IN[11] Pin/Port Type : Domain Boundary Single Load Supply : Yes Load Supply : (VDDL,VSS)

Level 2 -- REG_BLK/DATA_IN[11]_UPF_ISO_snps_pd3__iso_stra_snps_DATA_IN[11]_UPF_ISO/A Pin/Port Type : Load Pin Supplies : VDDL(Domain),VSS(Domain) Cell Name : REG_BLK/DATA_IN[11]_UPF_ISO_snps_pd3__iso_stra_snps_DATA_IN[11]_UPF_ISO Cell Type : Isolation Cell Power Domain : pd3 Library Cell : ISOLNX2M Associated Strategy : iso_stra(pd3)

Power Supplies -- Input -- -- Output -- VDDL(Domain Default) VDDL(Domain Default)

SEE ALSO

associate_mv_cells(2)
create_mv_cells(2)
map_isolation_cell(2)
map_level_shifter_cell(2)
set_isolation(2)
set_isolation_control(2)
set_level_shifter(2)
set_repeater(2)

report_name_rules

Reports the defined name rules.

SYNTAX

status **report_name_rules**
[*name_rules*]

Data Types

name_rules string

ARGUMENTS

name_rules

Specifies the name of the rules to be reported. If the *name_rules* argument is not specified, all currently defined name rules are reported. *name_rules* can be a GLOB style pattern also.

DESCRIPTION

This command reports the values of a set of name rules, that are created or modified by the *define_name_rules* command. This command generates a report for all the name rules matching the name pattern given as *name_rules* argument. When the *name_rules* argument is not specified, all the name rules are reported.

The report also contains the predefined special Verilog name rules. The Verilog name rules are predefined and can be modified using *define_name_rules* command. See the *define_name_rules* man page for more details on special rules.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the report_name_rules command to report the values of the name rules called EXAMPLE:

```
prompt> report_name_rules
****************************************
```

```
 Report     : name_rules
...
****************************************

Rules Name: EXAMPLE
   Collapse name space: false
   Equal port and net names: false
   Equal inout port and net names: false
   Check internal net name: false
   Target bus naming style: not defined
   Remove irregular port bus: false
   Check bus indexing: false
   Check bus indexing use type info: false
   Remove port bus: false
   Case insensitive: false
   Add dummy nets: false
   Reserved words: {in,inout,out}

   Object Max  Repl Rem
   Type   Len  Char Chrs Prefix Allowed Chars
   ------------------------------------------------------------
   Port   32   '_' no   P      Use "A-Z0-9"
                        First : Don't use "_"
                        Last : Don't use "_"
                        Mapping String: ""{"A","a"}""
   Cell   32   '_' no   U      Use "A-Z0-9"
                        First : Don't use "_"
                        Last : Don't use "_"
                        Mapping String: ""{"A","a"}""
   Net    32   '_' no   N      Use "A-Z0-9"
                        First : Don't use "_"
                        Last : Don't use "_"
                        Mapping String: ""{"A","a"}""
```

SEE ALSO

change_names(2)
define_name_rules(2)
report_names(2)

report_names

Reports the potential name changes if the specified rule is applied.

SYNTAX

status **report_names**
-rules *name_rules*
[-hierarchy]
[-include_sub_blocks]
[-skip_physical_only_cells]

Data Types

name_rules string

ARGUMENTS

-rules *name_rules*

Specifies a name rule set that details the rules for modifying names to which the object names conform. A name rule set is defined by using the **define_name_rules** command.

-hierarchy

Specifies that objects of all the logical hierarchies of the design are to be considered. By default, objects of top logical hierarchy are considered. This will not consider sub-blocks.

-include_sub_blocks

By default, objects of top logical hierarchy are considered. When *-hierarchy* option is used then all objects within current physical hierarchy is considered and it doesn't cross over into other physical hierarchies or sub-blocks. With this option specified, it will cross over into other physical hierarchies also and in doing so it will consider only the bound views. If an instance in top level is bound to non-design view of a sub-block like for example, abstract, then the command will error out. This is because design view can't be derived from a non-design view and changing non-design view will cause it to go out of sync with its design view. In order to consistently rename all the design data and associated timing constraints, only design views should be used for linking. This option can't be used independently and has to be used along with *-hierarchy*. If any of the sub-blocks is not yet linked then using this option will first link those sub-blocks and then change the names.

-skip_physical_only_cells

Specifies that physical-only cells should not be considered for name changes. This is applicable for nets and ports of these type of cells as well.

DESCRIPTION

The **report_names** command reports the names of ports, cells, and nets that would be changed to conform to the specified name rules. This command may not report all of the name changes that would occur by the **change_names** command.

The **report_names** command is normally run before the **change_names** command.

Use the **report_name_rules** command to display a list of name rules currently available. See the **define_name_rules** man page for information about naming rules that can be affected when running **report_names**.

With no options specified, **report_names** operates on ports, cells, and nets in the current design. When **-hierarchy** is specified, the report is expanded to include all design objects within the current design object hierarchy.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports port, cell, and net names in the current design that conform to the rules defined in default_name_rules:

```
prompt> report_names
****************************************
Report    : names - rules EXAMPLE
Version   : v 1.0
Date      : Fri Jul 31 20:30:42 2015
****************************************

Design   Type   Object          New Name
--------------------------------------------------------
TOP      port   sys_clk         SYS_CLK
TOP      port   sys_2x_clk       SYS_2X_CLK
TOP      cell   bufbda_G1B2I2    BUFBDA_G1B2I2
TOP      cell   bufbda_G1B1I8_2  BUFBDA_G1B1I8_2
TOP      cell   bufbda_G1B1I7    BUFBDA_G1B1I7
TOP      net    sdram_clk       SDRAM_CLK
TOP      net    scan_en         SCAN_EN
```

SEE ALSO

change_names(2)
define_name_rules(2)
report_name_rules(2)

report_net

Generates a report of net information.

SYNTAX

string **report_nets**
[-connections [-verbose]]
[-significant_digits *digits*]
[-segments]
[-physical]
[-nosplit]
[*net_list*]

list *net_names*

ARGUMENTS

-connections

Indicates that the report is to show net connection information.

-verbose

Must be used with the **-connections** option. Indicates that the report is to show verbose connection information.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default. Fixed-precision floating-point arithmetics is used for delay calculation; therefore, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h. The upper bound on a meaningful value of the argument to **-significant_digits** is then FLT_DIG - ceil(log10(fabs(*any_reported_value*))).

-segments

Indicates that the report is to show all global segments for each net requested. Global net segments are all those physically connected across all hierarchical boundaries.

-physical

Indicates that the report is to show the physical attributes and routing information for the net.

-nosplit

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing

software to extract information from the report output.

net_list

Specifies a list of nets to be reported. The default is to report all nets in the current instance or current design.

DESCRIPTION

The command displays information about the nets in the design of the current instance, or in the current design. If **current_instance** is set, the report is generated for the design of that instance; otherwise, the report is generated for the current design. Attributes, such as annotated resistance and annotated capacitance, are displayed.

If **-connections** is used, the report lists the leaf cell pins connected to each net.

If **-physical** is used, the report will list the physical attributes and routing information for the net.

EXAMPLES

The following example reports all nets in the design.

```
prompt> report_nets

****************************************
Report : net
 ...

****************************************

Attributes:
   c - annotated capacitance
   r - annotated resistance

Net             Fanout    Fanin      Cap   Resistance    Pins   Attributes
--------------------------------------------------------------------------------
a                  2        1     2.00        0.00      3
b                  1        1     1.00        0.00      2
c                  1        1     1.00        0.00      2
d                  1        1     1.00        0.00      2
e                  1        1     1.00        0.00      2
en                20        1    25.00        0.00     21
f                 1        1     1.00        0.00      2
g                  1        1     1.00        0.00      2
h                  1        1     1.00        0.00      2
i1                 1        1     1.00        0.00      2
i2                 1        1     1.00        0.00      2
i3                 1        1     1.00        0.00      2
i4                 1        1     1.00        0.00      2
j                 1        1     1.00        0.00      2
k                  1        1     1.00        0.00      2
l                 1        1     1.00        0.00      2
```

```
m                2       1     2.00       0.00      3
n                1       1     1.00       0.00      2
o                1       1     1.00       0.00      2
o1               1       1     0.00       0.00      2
o2               1       1     0.00       0.00      2
p                1       1     1.00       0.00      2
q                2       1     2.00       0.00      3
r                1       1     1.00       0.00      2
s                1       1     1.00       0.00      2
t                1       1     1.00       0.00      2
u                1       1     1.00       0.00      2
w                1       1     1.00       0.00      2
y                1       1     1.00       0.00      2
z                1       1     2.00       0.00      2
--------------------------------------------------------------------------------
Total 30 nets       52      30    56.00       0.00     82
Maximum             20       1    25.00       0.00     21
Average           1.73    1.00     1.87       0.00   2.73
```

The following example displays a verbose connection report for net z.

```
prompt> report_nets -verbose -connections z

****************************************
Report : net
        -connections
        -verbose
 ...

****************************************


Connections for net 'z':
   pin capacitance:   2
   wire capacitance:  0
   total capacitance: 2
   wire resistance:   0
   number of drivers: 1
   number of loads:   1
   number of pins:    2

   Driver Pins           Type              Pin Cap
   ------------          -----------------     --------
   z/Z                Output Pin (NOR2)     0

   Load Pins             Type              Pin Cap
   ------------          -----------------     --------
   o2/B               Input Pin (BUF)       2
```

The following example displays physical information for net ReadAdd[22].

```
prompt> report_nets -physical ReadAdd[22]

****************************************
```

```
Report : net
       -physical
 ...

****************************************

--------------------------------------------------------------------------------
Reporting for net 'ReadAdd[22]'
--------------------------------------------------------------------------------

Physical information:
------------------------------
Attribute Name      Value
------------------------------
flat_net          ReadAdd[22]
number_of_flat_pins 2
fanin             1
fanout            1
number_of_wires     4
number_of_vias      4
route_length        {M2 0.84} {M3 14.65} {M4 43.96}
max_layer_name
min_layer_name
physical_status     unrestricted
routing_rule        default

Routing information:
--------------------------------------------------------------------------------
Layer             Width       Start_X     Start_Y     End_X       End_Y
--------------------------------------------------------------------------------
M2                0.14        152.00      78.96       152.00      79.80
M3                0.14        152.00      79.80       153.44      79.80
M4                0.14        153.44      79.80       153.44      123.76
M3                0.14        153.44      123.76      166.65      123.76
VIA_S_307         *           152.00      78.96       *           *
VIA_S_306         *           152.00      79.80       *           *
VIA_S_305         *           153.44      79.80       *           *
VIA_S_304         *           153.44      123.76      *           *

Segments Information:
--------------------------------------------------------------------------------
Net Segment       is_tie_high  is_tie_low   net_type     number_of_pins
--------------------------------------------------------------------------------
ReadAdd[22]       false        false        signal       2
1
```

SEE ALSO

current_design(2)
current_instance(2)
report_cell(2)

report_design(2)

report_net_buses

Displays net bus information within the design.

SYNTAX

string **report_net_buses**
[-verbose]
[-nosplit]
[*port_bus_list*]

Data Types

net_bus_list string or collection

ARGUMENTS

-verbose

Includes in the output the name of each member net. By default, a summary section is displayed that lists all net buses, their start bit, end bit, and bit order.

-nosplit

Prevents line splitting if the column overflows. Most design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

net_bus_list

Displays information only on the specified nets buses in the current design. *net_bus_list* contains either a collection of net buses or a pattern that matches the net buses names.

DESCRIPTION

This command displays information about net buses in the current design. By default, the command produces a brief report including all net buses in the current design, their start bit, end bit, and bit order.

EXAMPLES

This example outputs the default net bus report.

```
prompt> report_net_buses

****************************************
Report : report_net_buses
Design : ORCA
Version:
Date   :
****************************************
Name          Start  End  Bit order
----------------------------------------
pad           15     0    down
sd_DQ         15     0    down
pc_be         3      0    down
sd_A          9      0    down
sd_BWS        1      0    down
1
```

SEE ALSO

get_net_buses(2)
remove_net_buses(2)
report_design(2)
report_hierarchy(2)
report_nets(2)
report_cells(2)
report_references(2)

report_net_estimation_rules

Writes out a report with information about net estimation rules.

SYNTAX

int **report_net_estimation_rules**
[-cell *cell*]
[*rules*]
cell collection

Data Types

rules string or list

ARGUMENTS

rules

Specifies one or more net estimation rules to report. The net estimation rule be previously define by the **set_net_estimation_rule** command. If no rule is specified, then the default rule is reported.

-cell *cell*

Specifies the physical cell from which to find the net estimation rule.

By default, the current block is searched for the specified net estimation rule(s)..

DESCRIPTION

This command reports net estimation rules as previously defined by the **set_net_estimation_rule** command.

NET ESTIMATION RULES AND HIERARCHY

Net estimation rules can be created in any physical block at any level of the hierarchy and multiple blocks can have different rules with the same name. However, rules are often stored/retrieved by name, and doing this can cause confusion as to which rule is being used by a given command. Generally, the "current" (top) block's list of rules is searched by commands such as *estimate_timing*. This means that if you define rule R1 in "top" and rule R1 in block "B1", if you make block B1 the current block, commands will use B1's R1. When you go back to top, top's R1 is used. To avoid confusion, use globally unique rule names and define them at the top - they will be propagated to child blocks during distributed commands as necessary. Rules in child blocks that have the same name as rules in top may also be overwritten during distributed commands.

EXAMPLES

The following example sets the register_spacing parameter and writes out a report for the default net estimation rule.

```
prompt> set_net_estimation_rule -parameter register_spacing -value 10 default
prompt> report_net_estimation_rules
****************************************
Report : report_net_estimation_rules
 ...

****************************************


rule name: default
 parameter              value(s)              value from
 ------------------------------------------------------------------
 layer                H: METAL4 V: METAL4      default
 utilization            H: 0.700 V: 0.700        default
 ndr                  N/A                    default
 xtalk_fraction         H: 0.000 V: 0.000        default
 corner               N/A
 ff_per_mm              H: N/A V: N/A
 ohms_per_mm            H: N/A V: N/A
 buffer               N/A
 buffer_spacing         H: N/A V: N/A
 delay_derate           1.000                  default
 ns_per_mm              H: N/A V: N/A
 clock                N/A
 sdc_stage_delay        N/A
 stage_margin           0.000                  default
 register             --ideal--              computed
 register_spacing       H: 10.000 V: 10.000      user
 launch_block_budget    N/A
 capture_block_budget   N/A
1
```

The following example writes out a net estimation rule report for rules *rule1* and *rule2*.

```
prompt> report_net_estimation_rules {rule1 rule2}
```

SEE ALSO

get_net_estimation_rules(2)
remove_net_estimation_rules(2)
set_net_estimation_rule(2)

report_net_fanout

Displays net fanout or buffer-tree information for the current design.

SYNTAX

status **report_net_fanout**
[-nosplit]
[-high_fanout]
[-threshold *lower*]
[-upper_bound *upper*]
[-connections]
[-physical]
[-tree [-depth *level*]]
[-hierarchical]
[*net_list*]

Data Types

lower integer
upper integer
level integer
net_list list

ARGUMENTS

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line in the same column.

-high_fanout

Specifies to show high fanout nets only. A high fanout net is a net with more fanouts than **500**. A high fanout buffer tree is a buffer tree with more leaf loads than the specified value. The fanouts cross hierarchies.

This option is mutually exclusive with the **-threshold** option.

-threshold *lower*

Specifies to only show nets with more fanout than *lower*. The fanouts cross hierarchies.

This option is mutually exclusive with the **-high_fanout** option.

-upper_bound *upper*

Specifies to only show nets with fanout less than or equal to *upper*. The fanouts cross hierarchies.

-connections

Displays information about pins connected to the nets.

-physical

Displays location information when applicable.

-tree

Indicates to treat a buffer tree as transparent. The leaf loads of the buffer tree are treated as the fanouts of the net. Hierarchical boundaries are not considered as leaf loads.

This option is mutually exclusive with the *net_list* argument.

-depth *level*

Displays only buffer trees with more levels than *level*. This option can only be used with the **-tree** option.

-hierarchical

Report fanout across physical hierarchies. If this option is not turned on, only the fanout in current physical hierarchy will be reported.

net_list

Specifies to process only those nets on the *net_list*. If *net_list* is not specified, all nets in the current instance are processed.

This argument is mutually exclusive with the **-tree** option.

DESCRIPTION

The **report_net_fanout** command displays net fanout or buffer-tree information in the current design. If a *net_list* is specified, the report is generated only for the specified nets.

All reports cross hierarchical boundaries as if the nets were flattened. This is consistent with the **report_net** command.

The **report_net_fanout** command can be used to find high-fanout nets, buffer trees, or long buffer chains in the current design or the current instance.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to find high-fanout nets:

```
prompt> report_net_fanout -high_fanout
```

The following example shows how to find high-fanout nets in a particular set of nets:

prompt> **report_net_fanout -high_fanout [get_nets scan*]**

The following example shows buffer trees with 10 leaf pins:

prompt> **report_net_fanout -tree -threshold 9 -upper_bound 10**

The following example shows how to find long buffer chains:

prompt> **report_net_fanout -tree -depth 8 -upper_bound 1**

SEE ALSO

create_buffer_trees(2)
current_design(2)
current_instance(2)
remove_buffer_trees(2)
report_buffer_trees(2)

report_net_weight_effort

Report net weight effort for coarse placement.

SYNTAX

status **report_net_weight_effort**
[-nets *collection of nets*]

ARGUMENTS

-nets *collection of nets*

Reports the net weight effort for all nets, or for the nets specified.

DESCRIPTION

The **report_net_weight_effort** command reports the net weight effort on nets for coarse placement. If no nets are specified, the net weight effort is reported for all nets.

EXAMPLES

The following example reports the net weight effort for the nets matching n12*.

```
prompt> report_net_weight_effort -nets [get_nets n12*]
Net          WeightEffort
n123          medium
n1238          high
n1239          none
n1236          medium
n1237          medium
n122          (default low)
n120          (default low)
n121          (default low)
```

SEE ALSO

set_net_weight_effort(2)
report_net_weight_effort(2)

report_nets

Generates a report of net information.

SYNTAX

string **report_nets**
[-connections [-verbose]]
[-significant_digits *digits*]
[-segments]
[-physical]
[-nosplit]
[*net_list*]

list *net_names*

ARGUMENTS

-connections

Indicates that the report is to show net connection information.

-verbose

Must be used with the **-connections** option. Indicates that the report is to show verbose connection information.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default. Fixed-precision floating-point arithmetics is used for delay calculation; therefore, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h. The upper bound on a meaningful value of the argument to **-significant_digits** is then FLT_DIG - ceil(log10(fabs(*any_reported_value*))).

-segments

Indicates that the report is to show all global segments for each net requested. Global net segments are all those physically connected across all hierarchical boundaries.

-physical

Indicates that the report is to show the physical attributes and routing information for the net.

-nosplit

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing

software to extract information from the report output.

net_list

Specifies a list of nets to be reported. The default is to report all nets in the current instance or current design.

DESCRIPTION

The command displays information about the nets in the design of the current instance, or in the current design. If **current_instance** is set, the report is generated for the design of that instance; otherwise, the report is generated for the current design. Attributes, such as annotated resistance and annotated capacitance, are displayed.

If **-connections** is used, the report lists the leaf cell pins connected to each net.

If **-physical** is used, the report will list the physical attributes and routing information for the net.

EXAMPLES

The following example reports all nets in the design.

```
prompt> report_nets

****************************************
Report : net
 ...

****************************************

Attributes:
   c - annotated capacitance
   r - annotated resistance

Net             Fanout    Fanin      Cap   Resistance    Pins   Attributes
-------------------------------------------------------------------------------
a                  2        1     2.00        0.00      3
b                  1        1     1.00        0.00      2
c                  1        1     1.00        0.00      2
d                  1        1     1.00        0.00      2
e                  1        1     1.00        0.00      2
en                 20       1    25.00        0.00     21
f                  1        1     1.00        0.00      2
g                  1        1     1.00        0.00      2
h                  1        1     1.00        0.00      2
i1                 1        1     1.00        0.00      2
i2                 1        1     1.00        0.00      2
i3                 1        1     1.00        0.00      2
i4                 1        1     1.00        0.00      2
j                  1        1     1.00        0.00      2
k                  1        1     1.00        0.00      2
l                  1        1     1.00        0.00      2
```

```
m                2       1     2.00       0.00      3
n                1       1     1.00       0.00      2
o                1       1     1.00       0.00      2
o1               1       1     0.00       0.00      2
o2               1       1     0.00       0.00      2
p                1       1     1.00       0.00      2
q                2       1     2.00       0.00      3
r                1       1     1.00       0.00      2
s                1       1     1.00       0.00      2
t                1       1     1.00       0.00      2
u                1       1     1.00       0.00      2
w                1       1     1.00       0.00      2
y                1       1     1.00       0.00      2
z                1       1     2.00       0.00      2
--------------------------------------------------------------------------------
Total 30 nets       52      30    56.00       0.00     82
Maximum             20       1    25.00       0.00     21
Average           1.73    1.00     1.87       0.00   2.73
```

The following example displays a verbose connection report for net z.

```
prompt> report_nets -verbose -connections z

****************************************
Report : net
        -connections
        -verbose
 ...

****************************************


Connections for net 'z':
   pin capacitance:   2
   wire capacitance:  0
   total capacitance: 2
   wire resistance:   0
   number of drivers: 1
   number of loads:   1
   number of pins:    2

   Driver Pins           Type              Pin Cap
   ------------          -----------------     --------
   z/Z                Output Pin (NOR2)     0

   Load Pins             Type              Pin Cap
   ------------          -----------------     --------
   o2/B               Input Pin (BUF)       2
```

The following example displays physical information for net ReadAdd[22].

```
prompt> report_nets -physical ReadAdd[22]

****************************************
```

```
Report : net
       -physical
 ...

****************************************

--------------------------------------------------------------------------------
Reporting for net 'ReadAdd[22]'
--------------------------------------------------------------------------------

Physical information:
------------------------------
Attribute Name      Value
------------------------------
flat_net          ReadAdd[22]
number_of_flat_pins 2
fanin             1
fanout            1
number_of_wires     4
number_of_vias      4
route_length        {M2 0.84} {M3 14.65} {M4 43.96}
max_layer_name
min_layer_name
physical_status     unrestricted
routing_rule        default

Routing information:
--------------------------------------------------------------------------------
Layer             Width       Start_X     Start_Y     End_X       End_Y
--------------------------------------------------------------------------------
M2                0.14        152.00      78.96       152.00      79.80
M3                0.14        152.00      79.80       153.44      79.80
M4                0.14        153.44      79.80       153.44      123.76
M3                0.14        153.44      123.76      166.65      123.76
VIA_S_307         *           152.00      78.96       *           *
VIA_S_306         *           152.00      79.80       *           *
VIA_S_305         *           153.44      79.80       *           *
VIA_S_304         *           153.44      123.76      *           *

Segments Information:
--------------------------------------------------------------------------------
Net Segment       is_tie_high  is_tie_low   net_type     number_of_pins
--------------------------------------------------------------------------------
ReadAdd[22]       false        false        signal       2
1
```

SEE ALSO

current_design(2)
current_instance(2)
report_cell(2)

report_design(2)

report_noise

Displays static noise information for the worst pins, or the specified pins, or all violators.

SYNTAX

status **report_noise**
[-mode *mode*]
[-corner *corner*]
[-scenario *scenario*]
[-nworst *worst_pin_count*]
[-all_violators]
[-significant_digits *digits*]
[-nosplit]
[*pin_list*]

Data Types

mode	collection
corner	collection
scenario	collection
worst_pin_count	int
digits	int
pin_list	list

ARGUMENTS

-mode *mode*

Specifies the mode to use for reporting. The command uses the scenario (if any) of the specified mode and the corner specified by the **-corner** option for reporting. If you do not specify any of the **-mode**, **-corner**, or **-scenario** options, the command uses data from the current scenario to generate the report. It is an error to specify this option with the **-scenario** option.

-corner *corner*

Specifies the corner to use for reporting. The command uses the scenario (if any) of the specified corner and the mode specified by the **-mode** option for reporting. If you do not specify any of the **-mode**, **-corner**, or **-scenario** options, the command uses data from the current scenario to generate the report. It is an error to specify this option with the **-scenario** option.

-scenario *scenario*

Specifies the scenario to use for reporting. If you do not specify any of the **-mode**, **-corner**, or **-scenario** options, the command uses data from the current scenario to generate the report. It is an error to specify this option with the **-mode** or the **-corner** options.

-nworst *worst_pin_count*

Specifies the number of load pins to be reported. The default value is 1.

-all_violators

Indicates that only violating pins (negative slack) are to be shown. If this option is used with the -nworst option, the number of violating pins will be limited by that value.

-significant_digits *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default.

-nosplit

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line splitting and facilitates writing software to extract information from the report output.

pin_list

Specifies a list of pins to report noise.

DESCRIPTION

The **report_noise** command displays the static noise information (width, height, and slack) for the worst pins or the specified list of pins. It computes "between rail noise," which is above low and below high noise. It uses slack type height. The si analysis should be enabled.

EXAMPLES

The following example reports the static noise values for the 5 worst pins.

prompt> **report_noise -nw 5 -sign 3**

****
Report : noise
Design : r4000
Version: M-2016.12-SP5-1-VAL
Date : Fri Sep 22 10:53:00 2017
****
noise_region: above_low
pin name width height slack

MemRead 0.017 0.330 -0.055
U309/I1 0.087 0.301 -0.026
Instruction_reg_14_/D 0.087 0.301 -0.026
Instruction_reg_15_/D 0.068 0.276 -0.001
U356/I1 0.068 0.276 -0.001
noise_region: below_high
pin name width height slack

MemRead 0.033 0.408 -0.133
alu/U184/A1 0.055 0.321 -0.046
U244/I0 0.055 0.318 -0.043
U309/I1 0.086 0.302 -0.027
Instruction_reg_14_/D 0.086 0.302 -0.027
1

SEE ALSO

si_enable_analysis(3)
shell.common.report_default_significant_digits(3)

report_ocvm

Displays information about advanced on-chip variation (OCV) or parametric on-chip variation (POCV) derate factor tables and coefficients, and displays details of the advanced OCV derate calculation.

SYNTAX

status **report_ocvm**
[-mode *mode*]
[-corner *corner*]
[-scenario *scenario*]
[-early]
[-late]
[-rise]
[-fall]
[-clock]
[-data]
[-cell_delay]
[-net_delay]
[-list_annotated]
[-list_not_annotated]
[-coefficient]
[-corner_sigma]
[-lib_cell]
[-min_depth]
[-nosplit]
[-significant_digits]
{-type aocvm|pocvm}
[*object_list*]

Data Types

mode	collection
corner	collection
scenario	collection
object_list	list

ARGUMENTS

-mode *mode*

Specifies the mode to use for reporting.

The command uses the scenario (if any) of the specified mode and the corner specified by the **-corner** option for reporting.

If you do not specify any of the **-mode**, **-corner**, or **-scenario** options, the command uses data from the current scenario to generate the report.

It is an error to specify this option with the **-scenario** option.

-corner ***corner***

Specifies the corner to use for reporting.

The command uses the scenario (if any) of the specified corner and the mode specified by the **-mode** option for reporting.

If you do not specify any of the **-mode**, **-corner**, or **-scenario** options, the command uses data from the current scenario to generate the report.

It is an error to specify this option with the **-scenario** option.

-scenario ***scenario***

Specifies the scenario to use for reporting.

If you do not specify any of the **-mode**, **-corner**, or **-scenario** options, the command uses data from the current scenario to generate the report.

It is an error to specify this option with the **-mode** or the **-corner** options.

-early

Indicates that only early AOCV/POCV derate tables are shown on the objects specified in the *object_list*.

-late

Indicates that only late AOCV/POCV derate tables are shown on the objects specified in the *object_list*.

-rise

Indicates that only rise AOCV/POCV derate tables are shown on the objects specified in the *object_list*.

-fall

Indicates that only fall AOCV/POCV derate tables are shown on the objects specified in the *object_list*.

-clock

Indicates that only clock AOCV/POCV derate tables are shown on the objects specified in the *object_list*.

-data

Indicates that only data AOCV/POCV derate tables are shown on the objects specified in the *object_list*.

-cell_delay

Indicates that only cell delay AOCV/POCV derate tables are shown on the objects specified in the *object_list*.

-net_delay

Indicates that only net delay AOCV/POCV derate tables are shown on the objects specified in the *object_list*.

-list_annotated

Indicates that leaf cells and global nets that are annotated with AOCV/POCV derate tables are listed.

-list_not_annotated

Indicates that leaf cells and global nets that are not annotated with AOCV/POCV derate tables are listed.

-coefficient

Shows the AOCV coefficients. You can specify a collection of lib_cell and leaf cell objects in the object_list to display coefficients annotated only on those objects.

-corner_sigma

Shows the POCV corner sigma value and POCV global sigma value(**time.pocvm_corner_sigma**).

-lib_cell

This option can be used with either the **-list_annotated** or **-list_not_annotated** options to report the library cell name instead of the instance name. Note that this report is design level which includes all corners.

-min_depth

Shows the AOCVM minimum depth.

-nosplit

Do not split lines when columns overflow.

-significant_digits *digits*

Specifies the number of significant digits (digits to the right of the decimal point) to report. Allowed values are from 0 through 13. The default is 2. Using this option overrides the value set by the **shell.common.report_default_significant_digits** application option.

-type aocvm|pocvm

Indicates the type of the analysis, and this is a required argument.

object_list

The object_list can be cells/nets/design/lib_cell objects and the command will report the corresponding AOCV/POCV derate tables for these objects. If a timing arc object is provided, the corresponding AOCV metrics and associated derates are printed for that arc. If the object_list is empty, a summary of the number of cells/nets that have AOCV/POCV derates is printed. For POCV, if a cell/lib_cell is provided, the command will report both POCV coefficients and whether LVF(slew/load) tables exist for related panes. Note that LVF tables for cells are corner level, and LVF tables for lib_cells are design level. When both POCV coefficients and LVF tables exist, the **time.pocvm_precedence** app option determines the precedence.

DESCRIPTION

This command reports user-specified information about advanced on-chip variation (AOCV) or parametric on-chip variation (POCV) analysis.

If the design has been annotated with AOCV/POCV derate tables using the **read_ocvm** command, the **report_ocvm** command outputs a summary showing the numbers of leaf cells and global nets annotated with AOCV/POCV derate tables. Lists of fully annotated, partially annotated, and non annotated leaf cells and global nets can be displayed using the **-list_annotated** and **-list_not_annotated** options.

EXAMPLES

In the following example, the design has been annotated with AOCV derate factors using the **read_ocvm** command.

```
prompt> report_ocvm -type aocvm
report_ocvm -type aocvm
****************************************
Report : aocvm
Design : top
Version: J-2014.06
****************************************
 Annotation |    Total    |    Full    |   Partial   |    None    |
-------------------------------------------------------------------------------
 Leaf cells |      28           28           0           0
    Nets    |      33           33           0           0
-------------------------------------------------------------------------------
            |      61           61           0           0
```

The following example shows the derate tables applied on an instance abufe3.

```
prompt> report_ocvm -type aocvm [get_cells abufe3]
report_ocvm -type aocvm [get_cells abufe3]
****************************************
Report : aocvm
Design : top
Version: J-2014.06
****************************************
 Distance | Depth
          | 1.00   2.00   3.00   4.00   5.00  10.00  50.00  100.00
-------------------------------------------------------------------
   --     0.70   0.71   0.72   0.73  0.74   0.75   0.76   0.77

Name                      Object    Process  Path_type   Sense
  Delay  Inherited
-------------------------------------------------------------------------
abufe3                    cell      early    clock       rise
  cell     --
abufe3                    cell      early    clock       fall
  cell     --
abufe3                    cell      early     data       rise
  cell     --
abufe3                    cell      early     data       fall
  cell     --

 Distance | Depth
          | 1.00   2.00   3.00   4.00   5.00  10.00  50.00  100.00
-------------------------------------------------------------------
   --     1.70   1.69   1.68   1.67   1.66   1.65   1.64   1.63

Name                      Object    Process  Path_type   Sense
  Delay  Inherited
-------------------------------------------------------------------------
abufe3                    cell      late     clock       rise
  cell     --
abufe3                    cell      late     clock       fall
  cell     --
abufe3                    cell      late      data       rise
  cell     --
abufe3                    cell      late      data       fall
```

```
  cell     --
```

The following example shows the arc metrics and the corresponding derates for a timing arc.

```
prompt> report_ocvm -type aocvm [get_timing_arcs \
  -from abufe3/I -to abufe3/Z]
report_ocvm -type aocvm [get_timing_arcs -from abufe3/I -to abufe3/Z]
****************************************
Report : aocvm
Design : top
Version: J-2014.06
****************************************
From-pin: abufe3/I
To-pin: abufe3/Z
Arc type: cell (clock network)

 AOCVM arc metrics   Launch     Capture
-------------------------------------------
Distance             123.40     112.60
Depth                11.00      13.00

 AOCVM arc derates   Launch     Capture
-------------------------------------------
early rise            0.75       0.85
early fall           0.75       0.85
late rise            1.65       1.45
late fall           1.65       1.45
```

The following example shows the lib_cells' POCV coefficients and LVF tables. Note that the test/BUF lib_cell only has LVF table in pane 1. This could happen when a lib_cell has an LVF table in one corner, but not in another corner.

```
prompt> report_ocvm -type pocvm \
  [get_lib_cells "test/DFF test/BUF test/MUX"]
****************************************
Report : ocvm -type pocvm
Design : top
Mode   : func
Corner : c1
Scenario: s1
Version: ...
Date   : ...
****************************************
Information: time.pocvm_precedence determines precedence between POCV
coefficient and slew/load tables.

POCV coefficient: 0.20

Name                    Object    Process   Voltage   Path_type
  Sense   Delay  Inherited
-------------------------------------------------------------------------------
test/BUF                lib_cell  early     *       clock
  rise   cell   test/BUF
test/BUF                lib_cell  early     *       clock
  fall   cell   test/BUF
test/BUF                lib_cell  early     *       data
  rise   cell   test/BUF
test/BUF                lib_cell  early     *       data
```

```
  fall   cell   test/BUF


POCV coefficient: 0.10

Name                    Object    Process   Voltage   Path_type
  Sense   Delay  Inherited
-------------------------------------------------------------------------------
test/BUF                lib_cell  late      *         clock
  rise   cell   test/BUF
test/BUF                lib_cell  late      *         clock
  fall   cell   test/BUF
test/BUF                lib_cell  late      *         data
  rise   cell   test/BUF
test/BUF                lib_cell  late      *         data
  fall   cell   test/BUF


Cells with POCV slew/load tables. Report the desired lib_arc to see the table.

Name                    Object    Pane
-----------------------------------------------------
test/DFF                lib_cell    0
test/DFF                lib_cell    1
test/BUF                lib_cell    1
test/MUX                lib_cell    0
test/MUX                lib_cell    1
```

The following example shows POCV annotations for the design broken down by lib_cells. The annotation type LVF will be reported if the lib_cell has LVF information in one corner, but not in another corner. You can get further information on this by using **get_lib_cell** with **report_ocvm**.

```
prompt> report_ocvm -type pocvm -lib_cell -list_annotated
****************************************
Report : ocvm -type pocvm
Design : top
Mode   : func
Corner : c1
Scenario: s1
Version: ...
Date   : ...
****************************************

POCV coefficient:

 Annotation |    Total    |    Full    |   Partial   |    None    |
-------------------------------------------------------------------------------
 Leaf cells |      34           30           0           4
   Nets     |      40           0            0           40
-------------------------------------------------------------------------------
            |      74           30           0           44

Cells annotated with POCVM coefficient factors:

Name                          Process  Sense    Path     Annotation Type
-------------------------------------------------------------------------------
test/DFF                        *        *        *      LVF
```

```
test/BUF                          *       *       *     LVF
test/MUX                          *       *       *     LVF

POCV distance:

 Annotation |     Total    |     Full     |   Partial   |     None    |
-------------------------------------------------------------------------
 Leaf cells |      34              0            0             34
   Nets     |      40              0            0             40
-------------------------------------------------------------------------
            |      74              0            0             74
```

SEE ALSO

read_ocvm(2)
remove_ocvm(2)
time.aocvm_enable_analysis(3)
time.pocvm_enable_analysis(3)

report_optimization_history

Report design optimization history.

SYNTAX

status **report_optimization_history**

DESCRIPTION

This command reports the history of the optimization steps applied to the design:

Current State: is mapped; is placed; Events: - initial_map 08/16/19 10:53 --> 0.001111 hours - logic_opto 08/16/19 10:53 --> 0.016667 hours - initial_place 08/16/19 10:54 --> 0.000833 hours - insert_dft 08/16/19 10:54 --> 0.000833 hours - initial_drc 08/16/19 10:54 --> 0.000833 hours

SEE ALSO

read_optimization_history(2)
write_optimization_history(2)

report_parasitic_parameters

Reports parasitic parameters for all constraint corners, or for only the specified corners.

SYNTAX

string **report_parasitic_parameters**
[-corners *corners*]

Data Types

corners list

ARGUMENTS

-corners *corners*

Specifies the corners for reporting parasitic parameters. If this option is not specified, the command reports parasitic parameters for all corners.

DESCRIPTION

This command reports parasitic parameters required by extraction for all corners. If you use the **-corners** option, the command reports only the specified corners.

Multicorner-Multimode Support

EXAMPLES

The following example reports the parasitic parameters for the current corner.

```
prompt> report_parasitic_parameters -corners [current_corner]
1
```

SEE ALSO

all_corners(2)
create_corner(2)
create_mode(2)
current_corner(2)
get_corners(2)
remove_corner(2)
report_extraction_options(2)
set_extraction_options(2)
set_parasitic_parameters(2)

report_parasitics

Displays the Arnoldi and/or Elmore delay calculations for an entire stage.

SYNTAX

status **report_parasitics**
[-mode *mode*]
[-corner *corner*]
[-scenario *scenario*]
[-early]
[-late]
[-rise]
[-fall]
[-xcap]
[-percentile]
[-significant_digits *digits*]
[*object_list*]

Data Types

mode collection
corner collection
scenario collection
digits integer
object_list collection

ARGUMENTS

-mode ***mode***

Specifies the mode for to be used for reporting. The scenario (if any) of the mode given by the **-mode** option and the corner specified by the **-corner** option will be used. If none of the **-mode**, **-corner**, or **-scenario** options are specified, the command reports using data from the current scenario. It is an error to give this option with the **-scenario** option.

-corner ***corner***

Specifies the corner for to be used for reporting. The scenario (if any) of the corner given by the **-corner** option and the mode specified by the **-mode** option will be used. If none of the **-corner**, **-mode**, or **-scenario** options are specified, the command reports using data from the current scenario. It is an error to give this option with the **-scenario** option.

-scenario ***scenario***

Specifies the scenario for to be used for reporting It is an error to give this option with the **-mode** or the **-corner** options. If none of the **-corner**, **-mode**, or **-scenario** options are specified, the command reports using data from the current scenario.

-late

Report using data for max (late) operating condition. If neither of the **-late** or **-early** options are given, max data will be reported. If both of these options are given, each will be reported separately.

-early

Report using data for min (early) operating condition. If neither of the **-late** or **-early** options are given, max data will be reported. If both of these options are given, each will be reported separately.

-rise

Report using pin capacitance data for rising signal transitions. If neither of the **-rise** or **-fall** options are given, rise data will be reported. If both of these options are given, each will be reported separately.

-fall

Report using pin capacitance data for falling signal transitions. If neither of the **-rise** or **-fall** options are given, rise data will be reported. If both of these options are given, each will be reported separately.

-xcap

Report details of cross-capacitance (if available).

-percentile

Print percentile instead of histogram statistics in the summary report.

-significant_digits ***digits***

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 through 13. The default is 2. This option overrides the value set by the **shell.common.report_default_significant_digits** application option.

object_list

Report the parasitics of the given nets, or the nets of the given pins or ports.

DESCRIPTION

The **report_parasitics** command provides a summary report of the parasitic RC networks, as well as a detailed description of the RC networks of the given objects.

The summary report includes information of the total and per-net average R and C values, as well as the statistical distribution of RC network size and total R/C.

The per-net detailed reports include the net's total parasitic resistance and capacitance, its total capacitance, and a detailed description of its RC network and each associated pin capacitance. If the **-xcap** option is given, cross-capacitors (if available) will be listed.

The **-early**, **-late**, **-rise**, and **-fall** options control which types of parasitics and pin capacitance are reported. Note that each specified min/max/rise/fall combination will be reported separately.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenario** option or the **-corner** and **-mode** options.

SEE ALSO

report_delay_calculation(2)
report_parasitic_parameters(2)
report_stage(2)

report_parasitics_derate

Resets parasitic derating factors that influence extraction for DR/GR/CTO/RDE.

SYNTAX

string **report_parasitics_derate**
[-corners *corners*]

Data Types

corners	list

ARGUMENTS

-corners *corners*

Specifies the list of corners for which to apply the extraction options. Reporting derating factors will for all corners without -corners.

DESCRIPTION

This command reports derating factors used during RC extraction. The derating options will be associated with default constraint corner, or each list of constraint corners.

Multicorner-Multimode Support

EXAMPLES

```
prompt> report_parasitics_derate

Global factors:
---------------

Corner: default
* max
  Cap factor(Data,Clock)        : {1.1, 1.3}
```

```
    Res factor(Data,Clock)        : {1.01, 1.02}
    Coupling factor(Data,Clock)   : {1.05, 1.1}

    Layer based scale
    -----------------
      Cap scale
        M3        : 0.98
        M5        : 1.01
        M1        : 0.99
  * min

    Layer based scale
    -----------------
      Cap scale
        M5        : 1.01
        M1        : 0.99
        M3        : 0.98
  1
```

SEE ALSO

all_corners(2)
create_corner(2)
create_mode(2)
current_corner(2)
get_corners(2)
remove_corners(2)
set_parasitics_derate(2)
reset_parasitics_derate(2)
report_parasitic_parameters(2)
set_parasitic_parameters(2)
get_user_units(2)

report_path_group

Reports path_group information.

SYNTAX

string **report_path_groups**
[-modes *mode_list*]
[-nosplit]
[*path_group_list*]

Data Types

mode_list list
path_group_list list

ARGUMENTS

-modes *mode_list*

Displays only the path groups associated with the *mode_list* modes.

-nosplit

Specifies not to split lines if a column overflows. Most of the information is displayed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line splitting and facilitates writing scripts to extract information from the report output.

path_group_list

Displays the path group information for the *path_group_list* that is specified.

DESCRIPTION

Produces a report showing information about path groups in the **current_design** command. The report includes path groups automatically created by the **create_clock** command and those manually created with the **group_path** command. Path groups are used to affect the calculation of the Maximum Delay cost during optimization. If no mode is specified on the command line, the path groups are reported for the **current_mode** command only; otherwise all path groups associated with the list of requested modes are displayed. If *path_group_list* is specified but contains an invalid group name, all path groups for the current mode will be displayed. The arguments *mode_list* and *path_group_list* cannot be used together.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-modes** option.

EXAMPLES

```
prompt> report_path_groups

****************************************
Report : path_group
 ...
****************************************

Path groups for mode 'M1':

Path_Group    Weight    From/Through/To
--------------------------------------------------------------------------------
**async_default**
          1.00     -
**clock_gating_default**
          1.00     -
**default**   1.00     -
clk          1.00     -to [get_clocks {clk}]


Path groups for mode 'M2':


Path_Group    Weight    From/Through/To
--------------------------------------------------------------------------------
**async_default**
          1.00     -
**clock_gating_default**
          1.00     -
**default**   1.00     -
Group1       1.00     -from [get_ports {clk clk_bis}]
  -to [get_cells {cptOut_reg[2] cptOut_reg[3]}]
                  -from [get_cells {cptOut_reg[0] cptOut_reg[1]}]
  -to [get_ports {Out[3] Out[2]}]
clk          1.00     -to [get_clocks {clk}]

prompt> current_mode M2
prompt> report_path_groups Group1

****************************************
Report : path_group
 ...
****************************************

Path groups for mode 'M2':

Path_Group    Weight    From/Through/To
--------------------------------------------------------------------------------
Group1       1.00     -from [get_ports {clk clk_bis}]
  -to [get_cells {cptOut_reg[2] cptOut_reg[3]}]
```

```
                -from [get_cells {cptOut_reg[0] cptOut_reg[1]}]
    -to [get_ports {Out[3] Out[2]}]
```

SEE ALSO

create_clock(2)
group_path(2)

report_path_groups

Reports path_group information.

SYNTAX

string **report_path_groups**
[-modes *mode_list*]
[-nosplit]
[*path_group_list*]

Data Types

mode_list list
path_group_list list

ARGUMENTS

-modes *mode_list*

Displays only the path groups associated with the *mode_list* modes.

-nosplit

Specifies not to split lines if a column overflows. Most of the information is displayed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line splitting and facilitates writing scripts to extract information from the report output.

path_group_list

Displays the path group information for the *path_group_list* that is specified.

DESCRIPTION

Produces a report showing information about path groups in the **current_design** command. The report includes path groups automatically created by the **create_clock** command and those manually created with the **group_path** command. Path groups are used to affect the calculation of the Maximum Delay cost during optimization. If no mode is specified on the command line, the path groups are reported for the **current_mode** command only; otherwise all path groups associated with the list of requested modes are displayed. If *path_group_list* is specified but contains an invalid group name, all path groups for the current mode will be displayed. The arguments *mode_list* and *path_group_list* cannot be used together.

Multicorner-Multimode Support

By default, this command works on the current mode. To specify a different mode, use the **-modes** option.

EXAMPLES

```
prompt> report_path_groups

****************************************
Report : path_group
 ...
****************************************

Path groups for mode 'M1':

Path_Group    Weight    From/Through/To
--------------------------------------------------------------------------------
**async_default**
          1.00     -
**clock_gating_default**
          1.00     -
**default**   1.00     -
clk          1.00     -to [get_clocks {clk}]


Path groups for mode 'M2':


Path_Group    Weight    From/Through/To
--------------------------------------------------------------------------------
**async_default**
          1.00     -
**clock_gating_default**
          1.00     -
**default**   1.00     -
Group1       1.00     -from [get_ports {clk clk_bis}]
  -to [get_cells {cptOut_reg[2] cptOut_reg[3]}]
                  -from [get_cells {cptOut_reg[0] cptOut_reg[1]}]
  -to [get_ports {Out[3] Out[2]}]
clk          1.00     -to [get_clocks {clk}]

prompt> current_mode M2
prompt> report_path_groups Group1

****************************************
Report : path_group
 ...
****************************************

Path groups for mode 'M2':

Path_Group    Weight    From/Through/To
--------------------------------------------------------------------------------
Group1       1.00     -from [get_ports {clk clk_bis}]
  -to [get_cells {cptOut_reg[2] cptOut_reg[3]}]
```

```
                -from [get_cells {cptOut_reg[0] cptOut_reg[1]}]
    -to [get_ports {Out[3] Out[2]}]
```

SEE ALSO

create_clock(2)
group_path(2)

report_pg_mask_constraints

Reports PG mask constraints. Specify a constraint name to limit the report to one PG mask constraint.

SYNTAX

status **report_pg_mask_constraints**
[*name*]
[-tcl]

Data Types

name string

ARGUMENTS

name

Reports the specified PG mask constraint. If this option is not specified, the command reports all PG mask constraints defined by **set_pg_mask_constraint**.

-tcl

Reports single specified PG mask constraint with Tcl format.

DESCRIPTION

This command reports the PG mask constraints. With no options, the command reports all mask constraints specified with the **set_pg_mask_constraint** command. If you specify a constraint name, the command reports only that constraint.

EXAMPLES

The following example reports the PG mask constraint named *r1*.

prompt> **report_pg_mask_constraints r1**

The following example reports the PG mask constraint named *r1* with Tcl format.

prompt> **report_pg_mask_constraints r1 -tcl**

SEE ALSO

set_pg_mask_constraint(2)
remove_pg_mask_constraints(2)

report_pg_patterns

Reports the specified power ground pattern.

SYNTAX

status **report_pg_patterns**
[*pattern_name*]
[-tcl]

Data Types

pattern_name string

ARGUMENTS

pattern_name

Reports the specified power ground pattern. If this option is not specified, the command reports all pre-defined power ground patterns.

-tcl

Reports single specified power ground pattern with Tcl format.

DESCRIPTION

This command reports the specified power ground pattern or all pre-defined patterns if the name is not specified.

EXAMPLES

The following example reports power ground wire pattern p1.

```
prompt> report_pg_patterns r1
Pattern Name    : p1
Pattern Type    : Wire
Direction       : Horizontal
Layer           : M3
Center          : 0
```

```
Low end        : 100
High end       : 300
Extend low     :
Extend high    :
Width          : 10
Spacing        : minimum
Trim           : true
Parameters     :
```

The following example reports power ground macro connection pattern r2.

```
prompt> report_pg_patterns p2
Pattern Name    : p2
Pattern Type    : Macro connection
Macro Pin Type  : Long pin
Nets            : pwr gnd
Direction       : Horizontal
Width           : @w
Layers          : M4
Spacing         : 3
Pitch           : 20
Number          :
Offset          : 3
Via rule        : {intersection: all}{via_master: VIA34}
Parameters      : w
```

The following example reports power ground macro connection pattern r2 with Tcl format.

```
prompt> report_pg_patterns p2 -tcl
```

SEE ALSO

create_pg_wire_pattern(2)
create_pg_ring_pattern(2)
create_pg_macro_conn_pattern(2)
create_pg_std_cell_conn_pattern(2)
create_pg_composite_pattern(2)
remove_pg_patterns(2)

report_pg_regions

Reports the specified power ground region names and areas.

SYNTAX

status **report_pg_regions**
[*region_name*]

Data Types

region_name string

ARGUMENTS

region_name

Reports the specified power ground region. If you do not specify a region name, the command reports all power ground regions.

DESCRIPTION

This command reports all power ground regions, or only the power ground region you specify.

EXAMPLES

The following example reports power ground region *r1*.

```
prompt> report_pg_regions r1
Region: r1
Points: {{100 150} {150 150} {150 100} {100 100}}
```

The following example reports all power ground regions.

```
prompt> report_pg_regions
Region: core_region
Points: {{225 374} {425 374} {425 225} {225 225}}
Region: r1
Points: {{100 150} {150 150} {150 100} {100 100}}
Region: r2
```

Points: {{200 250} {250 250} {250 200} {200 200}}

SEE ALSO

create_pg_region(2)
remove_pg_regions(2)

report_pg_strategies

Reports the specified power ground strategy.

SYNTAX

status **report_pg_strategies**
[*strategy_name*]
[-tcl]

Data Types

strategy_name string

ARGUMENTS

strategy_name

Reports the specified power ground strategy. If you do not specify a strategy name, the command reports all power ground strategies.

-tcl

Reports single specified power ground strategy with Tcl format.

DESCRIPTION

This command reports all power ground strategies, or only the specified power plan strategy.

EXAMPLES

The following example reports power ground strategy *s_mesh1*.

```
prompt> report_pg_strategies s_mesh1
name: s_mesh1
region:
   Region: core area
   Coordinate: {{400.0000 400.0000} {400.0000 3307.6080}
         {3308.2160 3307.6080} {3308.2160 400.0000} }
```

```
Pattern Expression :
   pattern: pg_mesh1
   nets: VDD - - VSS VSS - - VDD
   parameters:
   offset: undefined undefined
   offset start: 400.0000 400.0000
   pitch: undefined undefined
   repeat: undefined undefined

Instantiated Pattern :
Pattern Name: pg_mesh1
Type: Mesh Pattern
Parameters:
Layers:
{ {vertical_layer : M8} {width : {@w1 }} {spacing : interleaving}
  {trim : @t} {pitch : 20.672} {offset : 3.344} }
{ {horizontal_layer : M9} {width : {@w2 }} {spacing : interleaving}
  {trim : @t} {pitch : 26.752} {offset : 3.344} }
Via Rules:
{intersection: all} {via_master : {default}}

Extension:
   Nets:
   Layers:
   Sides:
   Direction: LRTB
   Stop: outer-most ring

Blockage:
```

The following example reports power ground strategy *s_mesh1* with Tcl format.

```
prompt> report_pg_strategies s_mesh1 -tcl
```

SEE ALSO

set_pg_strategy(2)
remove_pg_strategies(2)

report_pg_strategy_via_rules

Reports the specified power ground strategy via rule.

SYNTAX

status **report_pg_strategy_via_rules**
[*rule_name*]
[-tcl]

Data Types

rule_name string

ARGUMENTS

rule_name

Reports the specified power ground strategy via rule. If this option is not specified, the command reports all power ground strategy via rules defined by **set_pg_strategy_via_rule**.

-tcl

Reports single specified power ground strategy via rule with Tcl format.

DESCRIPTION

This command reports the specified power ground strategy via rule, or all pre-defined via rules by **set_pg_strategy_via_rule** if the name is not specified.

EXAMPLES

The following example reports power ground strategy via rule *r1*.

```
prompt> report_pg_strategy_via_rules r1
****************************************
Report : Power Ground Strategy Via Rule
Design : chip
Date   : Mon Feb 14 15:40:09 2010
```

```
****************************************
Via Rule Name   : r1
Via Rule        : {intersection: adjacent}{via_master: default}
```

The following example reports power ground strategy via rule *r1* with Tcl format.

```
prompt> report_pg_strategy_via_rules r1 -tcl
```

SEE ALSO

set_pg_strategy_via_rule(2)
remove_pg_strategy_via_rules(2)

report_pg_via_master_rules

Reports the specified power ground via rule.

SYNTAX

status **report_pg_via_master_rules**
[*rule_name*]
[-tcl]

Data Types

rule_name string

ARGUMENTS

rule_name

Reports the specified power ground via rule. If this option is not specified, the command reports all power ground via rules defined by **set_pg_via_master_rule**.

-tcl

Reports single specified power ground via rule with Tcl format.

DESCRIPTION

This command reports the specified power ground via rule, or all predefined via rules by **set_pg_via_master_rule** if the name is not specified.

EXAMPLES

The following example reports power ground via rule *r1*.

```
prompt> report_pg_via_master_rules r1
#----------------------------------------#
Via definition rule        : r1
Contact code               : VIA34
Offset                  : 3.0000 4.0000
```

```
    Array dimension (Column row)  : 5 5
    Pitches between arrays        : not set
    Spacing between cuts          : default default
    Offset start                  : center
    Orient                        : R0
    Via site ratio                : 1 1
    Track alignment               : no_align
    Cut mask                      : no_mask
    #-----------------------------------------#
```

The following example reports power ground via rule *r1* with Tcl format.

```
prompt> report_pg_via_master_rules r1 -tcl
```

SEE ALSO

set_pg_via_master_rule(2)
remove_pg_via_master_rules(2)

report_pin_blockages

report_pin_blockages

Reports the pin blockages in the design.

SYNTAX

status **report_pin_blockages**
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*pin_blockage_list*]

Data Types

digits int
pin_blockage_list list

ARGUMENTS

-verbose

Displays verbose pin blockage information, including the associated pins, nets, or ports.

-nosplit

Does not split lines if column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default.

pin_blockage_list

Specifies a list of pin blockages to report. The list can contain pin blockage names, patterns, or collections. A collection can be specified by using the **get_pin_blockages** command. This argument is mutually exclusive with the **-all** option. You must specify one, but not both.

DESCRIPTION

This command reports the user-specified pin blockage constraints in the design, as specified by the **create_pin_blockage** command. If you use the **-all** option, all pin blockages in the design, created by the **create_pin_blockage** command are reported.

The report includes pin blockage name, boundary, layers, and associated pins, nets, or ports.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the pin blockage *PIN_BLOCKAGE_1*:

```
prompt> report_pin_blockages -verbose PIN_BLOCKAGE_1

Pin Blockage       Description
-----------------------------------------------------------------
PIN_BLOCKAGE_1     Boundary:    (0.00 0.00}
                             (0.00, 2000.00)
                             (2000.00, 2000.00)
                             (2000.00, 0.00)
                   Layers:      metal1
                             metal2
                   Pins:        cellA/pin1
                             cellA/pin2
                             cellA/pin3
1
```

SEE ALSO

create_pin_blockage(2)
remove_pin_blockages(2)
get_pin_blockages(2)
add_to_pin_blockage(2)
remove_from_pin_blockage(2)
shell.common.report_default_significant_digits(3)

report_pin_buses

Displays pin bus information within the design.

SYNTAX

string **report_pin_buses**
[-verbose]
[-nosplit]
[*pin_bus_list*]

Data Types

pin_bus_list string or collection

ARGUMENTS

-verbose

Includes in the output the name of each member pin. By default, a summary section is displayed that lists all pin buses, their start bit, end bit, and bit order.

-nosplit

Prevents line splitting if the column overflows. Most design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

pin_bus_list

Displays information only on the specified pins buses in the current design. *pin_bus_list* contains either a collection of pin buses or a pattern that matches the pin buses names.

DESCRIPTION

This command displays information about pin buses in the current design. By default, the command produces a brief report including all pin buses in the current design, their start bit, end bit, and bit order.

EXAMPLES

This example outputs the default pin bus report.

```
prompt> report_pin_buses

****************************************
Report : report_pin_buses
Design : ORCA
Version:
Date   :
********************************************
Name                Start  End  Bit order
------------------------------------------------
cellBus/bus_A         9     0   down
cellBus/bus_B         0     7   up
1
```

SEE ALSO

create_pin_bus(2)
get_pin_buses(2)
remove_pin_buses(2)
report_design(2)
report_hierarchy(2)
report_references(2)

report_pin_constraints

Generates a report of pin constraints.

SYNTAX

string **report_pin_constraints**
[-nosplit]
[*pin_constraints_list*]
[-pins *pins*]
[-ports *ports*]
[-nets *nets*]

Data Types

pin_constraints_list collection
pins collection
ports collection
nets collection

ARGUMENTS

-nosplit

If the information for a given pin constraint exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing software to extract information from the report output. The **-nosplit** option is mutually exclusive with the **-pins**, **-ports**, and **-nets** options.

pin_constraints_list

Specifies a list of pin constraint objects to report. This collection may include only individual and/or bundle pin constraints. The **pin_constraints_list** option is mutually exclusive with the **-pins**, **-ports**, and **-nets** options.

-pins *pin_collection*

Specifies the pins to report. The **-pins** option is mutually exclusive with the **pin_constraints_list**, **-nosplit**, **-ports**, and **-nets** options.

-ports *port_collection*

Specifies the ports to report. The **-ports** option is mutually exclusive with the **pin_constraints_list**, **-nosplit**, **-pins**, and **-nets** options.

-nets *net_collection*

Specifies the nets to report. The **-nets** option is mutually exclusive with the **pin_constraints_list**, **-nosplit**, **-ports**, and **-pins** options.

DESCRIPTION

This command generates a report describing the pin constraints in the current design. The report includes basic attribute information for each constraint. By default, the command reports all individual and bundle pin constraint in the current design, unless the user specifies specific objects using the **pin_constraints_list** argument or one of the **-pins**, **-ports**, or **-nets** options.

The **-pins**, **-ports**, or **-nets** options report all related constraints, including individual pin constraints, block pin constraints, bundle pin constraints, and topological pin constraints on every level set by using a related set or create constraint command.

The **-pins**, **-ports**, or **-nets** options are mutually exclusive with the **pin_constraints_list** and **-no_split** options.

EXAMPLES

The following example displays all related constraints for the I_ALU/Optr pin.

```
prompt> report_pin_constraints -pins [get_pins I_ALU/Optr]
```

SEE ALSO

report_individual_pin_constraints(2)
report_block_pin_constraints(2)
report_bundle_pin_constraints(2)
report_topological_constraints(2)

report_pin_guides

report_pin_guides

Reports the pin guides in the design.

SYNTAX

status **report_pin_guides**
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*pin_guide_list*]

Data Types

digits int
pin_guide_list list

ARGUMENTS

-verbose

Displays verbose pin guide information, including the associated pins, nets, or ports.

-nosplit

Retains wide lines and does not split lines if column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default.

pin_guide_list

Specifies a list of pin guides to report. The list can contain pin guide names, patterns, or collections. A collection can be specified by using the **get_pin_guides** command. This argument is mutually exclusive with the **-all** option. You must specify one, but not both.

DESCRIPTION

This command reports the user-specified pin guide constraints in the design, as specified by the **create_pin_guide** command. If you use the **-all** option, all pin guides in the design, created by the **create_pin_guide** command are reported.

The report includes pin guide name, boundary, layers, parent modules, exclusivity, pin spacing, and associated pins, nets, or ports.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports the pin guide *PIN_GUIDE_1*:

```
prompt> report_pin_guides -verbose PIN_GUIDE_1

Pin Guide         Description
-----------------------------------------------------------------
PIN_GUIDE_1       Boundary:    (0.00 0.00}
                             (0.00, 2000.00)
                             (2000.00, 2000.00)
                             (2000.00, 0.00)
                  Layers:      metal1
                               metal2
                  Parents:     cellA
                  Exclusive:   false
                  Pin Spacing: not specified
                  Pins:        cellA/pin1
                               cellA/pin2
                               cellA/pin3
1
```

SEE ALSO

create_pin_guide(2)
remove_pin_guides(2)
get_pin_guides(2)
add_to_pin_guide(2)
remove_from_pin_guide(2)
shell.common.report_default_significant_digits(3)

report_pin_name_synonym

Reports pin-name synonym definitions.

SYNTAX

status **report_pin_name_synonym**
[-nosplit]

ARGUMENTS

-nosplit

Prevents line splitting. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The report_pin_name_synonym command displays all of the currently defined pin name synonyms.

EXAMPLES

The following command set shows the pin-name synonym settings and then uses the report_pin_name_synonym command to report pin-name synonyms:

```
prompt> set_pin_name_synonym SYN_CD CD
1

prompt> set_pin_name_synonym SYN_QN QN
1

prompt> set_pin_name_synonym -full_name \
    mid1/bot1/LSR0P_at_bot/SYN_R mid1/bot1/LSR0P_at_bot/R
1

prompt> set_pin_name_synonym -full_name this/is/a/synonym \
    mid1/FJK2SP_at_mid/TI
1
```

```
prompt> set_pin_name_synonym -full_name mid1/FJK2SP_at_mid/J \
      mid2/bot2/LSR0P_at_bot/Q
1

prompt> report_pin_name_synonym
****************************************
Report : pin name synonym
Design : top
Version: X-2005.09
Date   : Wed Jun 29 10:29:04 2005
****************************************

Synonym                    Pin Name                  Type
--------------------------------------------------------------------------------
this/is/a/synonym              mid1/FJK2SP_at_mid/TI        full name
mid1/FJK2SP_at_mid/J           mid2/bot2/LSR0P_at_bot/Q     full name
mid1/bot1/LSR0P_at_bot/SYN_R   mid1/bot1/LSR0P_at_bot/R     full name
SYN_QN                     QN                        simple name
SYN_CD                     CD                        simple name
--------------------------------------------------------------------------------
1
```

SEE ALSO

remove_pin_name_synonym(2)
set_pin_name_synonym(2)

report_pin_placement

Reports the pin name, layer, side and offset for block pins or top-level terminals in the design.

SYNTAX

string **report_pin_placement**
[-cells *cells*]
[-nets *nets*]
[-pins *pins*]
[-ports *ports*]
[-self]
[-format {layer side offset}]

Data Types

cells collection
nets collection
pins collection
ports collection

ARGUMENTS

-cells *cells*

Specifies the block cells for which to report pin locations. If this option is not specified, the command reports pin locations for all block cells.

-nets *nets*

Specifies the nets for which to report pin locations. If this option is not specified, the command reports pin locations for all nets.

-pins *pins*

Specifies the pins for which to report pin locations. If this option is not specified, the command reports pin locations for all block pins.

-ports *ports*

Specifies the top-level terminals for which to report pin locations. If this option is not specified, the command reports pin locations for all top-level terminals.

-self

Reports output terminal locations for the top-level design. If this option is not specified and the **-cells**, **-nets**, **-pins** and **-ports** options are not specified, the command reports pin locations for all child block cells.

-format {layer side offset}

Limits the report to the specified information. The **-format {layer}** option reports only layer information. The **-format {layer side}** option reports both layer and side. Side is automatically included when you specify offset. By default, the command reports layer, side, and offset for each pin.

The side specifies the side number for a reference block where the pins are inserted. The side number is a positive integer that starts from 1. Given any block with a rectangular or rectilinear shape, the leftmost edge is side number 1. If there are multiple leftmost edges, then edge 1 is the lowest leftmost edge. The sides are numbered in consecutive order proceeded clockwise around the shape. The shape here referring to block

The offset is calculated based on the center of the corresponding pin. The offset specifies the distance in microns from the starting point of a specific side to the block pin's location on that side in clockwise direction. The starting point also follows the clockwise ordering. For instance, for a bottom horizontal side, the starting point is the right vertex of the side as it is counted in clockwise and bottom side's starting point is right point vertex. The side refers to the side of the reference block.

DESCRIPTION

This command reports the cell, layer, side, offset for each block pin or top-level terminal in the design. Use command options to limit to report to specific blocks, nets, pins, or ports. Use the **-self** option to report the placement settings for top-level terminals. Blocks, nets and pins on every level can be reported.

EXAMPLES

The following example writes a report to the terminal that contains the pin layer, side and offset information for pin A.

```
prompt> report_pin_placement -pins [get_pins A] -format {layer side offset}
```

The following example writes a report for all block pins to the pins.rpt file in the local directory.

```
prompt> report_pin_placement > pins.rpt
```

The following example writes a report for all block pins to the pins.rpt file in the local directory. The report contains only the layer for each block pin.

```
prompt> report_pin_placement -format {layer} > pins.rpt
```

SEE ALSO

check_pin_placement(2)
place_pins(2)

report_pipeline_scan_data_configuration

Displays options specified by the **set_pipeline_scan_data_configuration** command.

SYNTAX

status **report_pipeline_scan_data_configuration**

ARGUMENTS

The **report_pipeline_scan_data_configuration** command has no arguments.

DESCRIPTION

This command displays options specified by the **set_pipeline_scan_data_configuration** command on the current design.

The command reports the following information:

- Head Pipeline Clock
- Tail Pipeline Clock
- Head Pipeline Stages
- Tail Pipeline Stages

EXAMPLES

The following is an example of a pipeline scan data configuration report:

```
prompt> report_pipeline_scan_data_configuration
****************************************
Report : Pipeline Specification
Design : top
Version: P-2019.03
Date   : Thu Feb  7 07:07:54 2019
****************************************
```

```
Head Clock Name          PCLK
Tail ClockName           PCLK
Head Pipeline Stage      2
Tail Pipeline Stage      2
```

SEE ALSO

set_pipeline_scan_data_configuration(2)

report_placement

Generates a quality of results (QoR) report for placement, including hard macro placement and standard cell placement.

SYNTAX

status **report_placement**
[-physical_hierarchy_violations all | internal | external | none]
[-voltage_area_violations all | internal | external | none]
[-wirelength all | interface | hard_macro | IO | none]
[-swimming_pool_area]
[-thin_channel_area]
[-hard_macro_route_over]
[-hard_macro_overlap]
[-hard_macro_hierarchy_perimeter]
[-hierarchical]
[-ignore_fixed]
[-error_view *view_name*]
[-verbose minimum | low | high]
[-hard_macro_pin_track_violations all | constrained_only | none]
[-hard_macro_orientation_violations]
[-hard_macro_density_gradient_violations]
[-user_grid]
[-poly_rule]
[-macro_spacing_rule]
[-maximum_swimming_pool_area *value*]
[-maximum_thin_channel_width *value*]
[-maximum_thin_channel_height *value*]

Data Types

view_name string
value float

ARGUMENTS

-physical_hierarchy_violations all | **internal** | **external** | **none**

Specifies the type of placement violations to report. An **internal** violation is caused by a cell that belongs to a block and is placed outside its core area. An **external** violation is caused by a cell that does not belong to a block, but is placed inside the block boundary. **all** reports both internal and external violations. **none** skips physical hierarchy violations. By default, the reporting type is **all**. It is expected that the placer occasionally leaves small, single row height violations on standard cells, so these are not reported. It is expected the legalizer can handle these.

-voltage_area_violations all | **internal** | **external** | **none**

Specifies the type of voltage area macro placement violations to report. An **internal** violation is caused by a cell that belongs to a voltage area which is associated with, but not placed inside it. An **external** violation is caused by a cell that does not belong to a voltage area, but is placed inside the voltage area. **all** reports both internal and external violations. **none** skips voltage area placement violations. By default, the reporting type is **internal**. This command option does not care the legality of standard cell voltage area placement.

-wirelength all | **interface** | **hard_macro** | **IO** | **none**

Specifies which nets to consider when reporting the half perimeter bounding box wirelength of the design. **all** specifies all nets. **interface** specifies nets that cross a physical boundary. **hard_macro** specifies nets that are connected only to hard macros. **IO** specifies nets connected only to I/Os (ports or pads). **none** skips wirelength violations. By default, the reporting type is **all**.

By default, the command reports two wirelength values for the top-level design. The first value is reported by stopping at the subblock pins. If the pin is not assigned, then the closest location for the pin is used. The second, larger wirelength (aka see-through wirelength) value considers the connectivity through the subblocks for the interface nets. If the "-hierarchical" option is used, then these two wirelengths are also reported for subblocks. However, one difference is that the second wirelength for subblocks does not consider subblock port connectivity. This is because that wirelength is already accounted for at the top level.

-swimming_pool_area

Reports the total area in design which is enclosed (or almost enclosed) by blocks, hard macros, soft placement blockages or hard placement blockages. These areas can lead to placement or routing problems and should be covered with a placement blockage or removed.

-thin_channel_area

Reports the total area in design which are thin channels between hard macros (also blocks). Thin channels can lead to placement or routing problems and should be widened or covered with a placement blockage.

-hard_macro_route_over

Reports an estimate of the number of nets that route over the hard macros in the design. Horizontal and vertical routes are reported separately.

-hard_macro_overlap

Reports on overlaps between hard macros and also between hard macros and hard macro blockages. This report will consider any user-defined hard macro keepout margin.

-hard_macro_hierarchy_perimeter

Reports the sum of perimeters for hard macros at the parent level and the grandparent level. The first value reported is the sum of bounding box perimeters for hard macros that belong to the parent hierarchy. The second value reported is the sum of bounding box perimeters for hard macros that belong to the grandparent hierarchy. This option reports the sum of hard macro hierarchy perimeters for all parent and grandparent hierarchies in the design that have at least two hard macros.

-hierarchical

Reports wirelength and violations at the top level and for subblocks in the design. Subblock reports are not generated for I/O wirelength and hard macro route-overs. By default, the command reports only on the current level of physical hierarchy.

-ignore_fixed

Controls whether to skip violations between fixed hard macros and all other hard macros. This only affects the **-physical_hierarchy_violations**, **-hard_macro_overlap** and **-hard_macro_density_gradient_violations** options. This option is typically used for certain special hard macros where overlaps are allowed and are manually created.

-error_view *view_name*

Specifies the name of the file in which to save the generated error view. If this option is not used, no separate error view file is created.

-verbose minimum | low | high

Controls the relative number of messages generated by the command. **minimum** results in the minimal amount of information. **low** results in a one or two page report. **high** indicates no limit to report size. By default, the relative number of messages is **low**.

-hard_macro_pin_track_violations all | constrained_only | none

Reports the sum of illegal pin shapes which are misaligned or color mismatched with wire tracks in hard macros. The values for the parameter are: "all", "constrained_only" and "none". The value "all" means all hard macros in design are checked. The default parameter value is "none" to be backward compatible. Only macros having constraints set with command **set_macro_constraints -align_pins_to_tracks list_of_macros** are considered in "constrained_only" mode for consistency with command **create_placement -floorplan**.

For a legal pin shape aligned with wire track, it meets following two requirements:

(1) The color attribute matched between pin shape and wire track. The following table shows how to judge the color attribute of pin shape and wire track is matched or not:

|--|
| pin_shape| | | | |
track	mask_one	mask_two	no_mask	same_mask
no_mask	no	no	yes	warning
mask_one	yes	no	yes	warning
mask_two	no	yes	yes	warning

If the color attribute of pin shape is not set, tool will recognize its color attribute as no_mask. If the color attribute of wire track is not set, tool will recognize its color attribute as no_mask.

(2) The track center is aligned with pin shape. Without option **-verbose**, this command only reports the sum of illegal pin shapes in hard macros. With the option **-verbose high**, more detailed information for each illegal pin shape is reported.

Pin shape size (width for vertical layer and height for horizontal layer) is larger than the threshold value specified using application option **plan.macro.pin_shape_track_match_threshold** will be ignored in the checking.

The application option **plan.macro.align_pins_on_parallel_edges** controls aligning pin shape along macro edge in the direction parallel to that macro edge. By default, pins are expected to be close to an edge, and to align in the direction perpendicular to that edge.

-hard_macro_orientation_violations

Report the orientation violations of hard macros in the design. The current orientation of a hard macro is considered as its orientation relative to the orientation of its parent block. When the current orientation of the hard macro is not one of its allowed orientations, neither its legal orientations, the report will list the name of the hard macro, its current orientation, allowed orientations and legal orientations. The allowed orientations could be set using command **set_macro_constraints -allowed_orientations**, and the legal orientations could be set using command **set_attribute -name allowable_orientation**. The report will also show the number of cells with orientation violations in each design and the total number of violations.

-hard_macro_density_gradient_violations

Reports the density gradient violations for all hard macros in all layers. This option calculates the density of inner window (Din) and outer window (Dout). If |Din - Dout| >= gradient tolerance, it will report this violation. The parameters used in checking hard macro density gradient violations are specified by the **set_density_gradient_options** command.

-user_grid

Reports violations of the user_grid constraint set on hard macros. The grid can be created by using the **create_grid** command.

And the constraint can be set using command **set_macro_constraints** with the **-alignment_grid** option.

-poly_rule

Reports poly rule violations. The poly rule requires that in a group of continuous poly stripes with the same length, the number of poly stripes must be greater than or equal to N. If the number of poly stripes is less than N, then it must have poly stripes greater than or equal to N on both right and left sides. Here N is the narrow placeable site threshold and can be specified by the **plan.place.poly_rule** application option.

-macro_spacing_rule

Reports on macro spacing rule violations between hard macros. This macro spacing rule is set with the **plan.macro.spacing_rule_widths** and **plan.macro.spacing_rule_heights** application options.

-maximum_swimming_pool_area *value*

Report swimming pools whose area is smaller than the threshold value.

-maximum_thin_channel_width *value*

Report thin channels whose width is smaller than the threshold value. For thin channels whose height is smaller than default threshold that tool computes are also reported.

-maximum_thin_channel_height *value*

Report thin channels whose height is smaller than the threshold value. For thin channels whose width is smaller than default threshold that tool computes are also reported.

DESCRIPTION

This command generates a placement report to indicate the quality of placement. By default, the command reports on the total half perimeter bounding box wirelength in the design and the placement violations in the design. Other reports are available, based on the command options you provide.

EXAMPLES

Here is an example output.

```
prompt> report_placement -hard_macro_overlap

****************************************
Report : report_placement
Design : orca
Version:
Date   : Wed Jul 10 16:09:13 2013
****************************************

  Wire length report (all)
  ==================
  wire length in design orca: 24001162.451 microns.
    number of nets with unassigned pins: 10
  wire length in design orca (see through blk pins): 2405463.472 microns.
```

```
  Physical hierarchy violations report
  ====================================
  Violations in design orca:
Information: cell rselector_2/U240 overlaps block B4 (DPP-403)
Information: cell rselector_5/rs_pointer_even_reg_2_ overlaps block B8 (DPP-403)
    0 cells placed outside of placement area.
    2 cells placed overlapping a sub-block.

  Hard macro to hard macro overlap report
  =======================================
Information: cont_ram/attribute_table and cont_ram/vq_head_table overlap (DPP-405)
Information: cont_ram/attribute_table and cont_ram/vq_tail_table overlap (DPP-405)
Information: fe_4/s_table and cont_ram/attr_table overlap (DPP-405)
Information: fe_3/s_table and cont_ram/attr_table overlap (DPP-405)
  HM to HM overlaps in design orca: 8

prompt> report_placement -hard_macro_pin_track_violations all -verbose high

****************************************
Report : report_placement
Design : orca
Version:
Date   :
****************************************

  Wire length report (all)
  ========================
  wire length in design orca: 24001162.451 microns.
    number of nets with unassigned pins: 10
  wire length in design orca (see through blk pins): 2405463.472 microns.
Report illegal macro placement base on pin-track attribute matching rule.
================================================================================
The number of illegal pin shapes of test_inst1 :       34
================================================================================
Pin shape     Wire track  Pin shape mask  Wire track color   center_aligned
L0          TRACK_22    mask_one        mask_one          NO
L1          TRACK_22    mask_one        mask_one          NO
L2          TRACK_22    mask_one        mask_one          NO
L3          TRACK_22    mask_one        mask_one          NO
L4          TRACK_22    mask_one        mask_one          NO
L5          TRACK_22    mask_one        mask_one          NO
================================================================================
```

SEE ALSO

create_keepout_margin(2)
create_placement(2)
create_placement_blockage(2)
report_density_gradient_options(2)
set_density_gradient_options(2)
plan.macro.pin_shape_track_match_threshold(3)
plan.macro.align_pins_on_parallel_edges(3)

report_placement_attractions

Reports placement_attractions in the current design.

SYNTAX

int **report_placement_attractions**
[-verbose]
[-nosplit]
[*placement_attraction_list*]

Data Types

digits int
placement_attraction_list list

ARGUMENTS

-verbose

Displays verbose placement_attraction information.

-nosplit

Does not split lines if column overflows.

placement_attraction_list

Specifies a list of placement_attractions to report. The list may contain bound names, patterns, or collections. A collection may be specified by using the **get_placement_attractions** command.

DESCRIPTION

The **report_placement_attractions** command generates a report about the bound constraints in the design. The report includes the placement_attraction name and a description of the placement_attraction. If **-verbose** is specified, the report also includes placement_attraction region coordinates, dimensions, and cells and ports assigned to the placement_attraction.

If *placement_attraction_list* is specified, those bounds will be reported. If it is not specified, then all placement_attractions in the design will be reported.

EXAMPLES

The following example reports all placement_attractions in the design.

```
prompt> report_placement_attractions *
Placement Attraction          Description
-------------------------------------------------------------------------
attraction1                   Add internal logic: false
                              Effort:          medium
                              Exclude buffers:   edge
                              Region:          floating
1
```

SEE ALSO

create_placement_attraction(2)
get_placement_attractions(2)
add_to_placement_attraction(2)
remove_from_placement_attraction(2)
remove_placement_attractions(2)

report_placement_ir_drop_target

Report IR drop voltage area targets.

SYNTAX

string **report_placement_ir_drop_target**

ARGUMENTS

DESCRIPTION

This command will show all IR aware placement target settings for all voltage areas in the design. The report has three columns: the voltage area, the low target and the high target. The two target columns are subdivided into a percentage and its type columnn. For each voltage area a row is printed. The number in the percentage column is target number in percentages. The type denotes the type of target: **vdrop** stands for a voltage drop target; **cells** stands for a population target and **default** shows that the target is not set and the default is used. The default for the low target is defined by app option **place.coarse.ir_drop_target_voltage_drop_low_percentage**. For the high target it is defined by app option **place.coarse.ir_drop_target_voltage_drop_high_percentage**.

EXAMPLES

The following example reports the voltage drop settings:

```
prompt> set_placement_ir_drop_target DEFAULT_VA high 2.0
prompt> get_placement_ir_drop_target DEFAULT_VA
{ { DEFAULT_VA high 2.000000 } }
prompt> report_placement_ir_drop_target

Placer: IR aware voltage drop target settings:
Voltage area --target low- -target high-
          vdrop    type vdrop    type
------------- ----- ------- ----- -------
 DEFAULT_VA   1.0 default   2.0   cells

1
```

SEE ALSO

set_placement_ir_drop_target(2)
get_placement_ir_drop_target(2)
reset_placement_ir_drop_target(2)
place.coarse.ir_drop_default_target_low_population(3)
place.coarse.ir_drop_default_target_low_percentage(3)
place.coarse.ir_drop_default_target_high_population(3)
place.coarse.ir_drop_default_target_high_percentage(3)

report_placement_spacing_rules

Generates a report of placement spacing rules in the current design as set by commands **set_placement_spacing_label** and **set_placement_spacing_rule**.

SYNTAX

status **report_placement_spacing_rules**

DESCRIPTION

This command generates a placement spacing rule report. It lists

- All placement spacing labels
- The library cells to which they apply, on which side of the library cell in the R0 orientation
- All rules that apply to the given labels

EXAMPLES

The following example reports the placement spacing rules.

```
prompt> report_placement_spacing_rules
------------------------------------------------------
    Placement Spacing Labels:

    Label   side    lib_cell
    SNPS_BOUNDARY
    A       left    tcbn90glvt/AN2LVTD8
                    tcbn90ghvt/AO222HVTD1
                    tcbn90glvt/AO222LVTD0
                    tcbn90ghvt/AOI22HVTD0
                    tcbn90ghvt/BUFFHVTD0
                    tcbn90ghvt/BUFFHVTD3
                    tcbn90glvt/BUFFLVTD16
                    tcbn90glvt/BUFFLVTD2
                    tcbn90glvt/BUFFLVTD4
            right
    B       left
            right   tcbn90glvt/AN2LVTD8
                    tcbn90ghvt/AO222HVTD1
```

```
                tcbn90glvt/AO222LVTD0
                tcbn90ghvt/AOI22HVTD0
                tcbn90ghvt/BUFFHVTD0
                tcbn90ghvt/BUFFHVTD3
                tcbn90glvt/BUFFLVTD16
                tcbn90glvt/BUFFLVTD2
                tcbn90glvt/BUFFLVTD4
------------------------------------------------------------
    Placement Spacing Rules:

    Label   Label   Illegal spacing
    A       B       0-3
------------------------------------------------------------
1
```

SEE ALSO

set_placement_spacing_label(2)
set_placement_spacing_rule(2)
remove_placement_spacing_rules(2)

report_pop_up_object_options

Reports options for command pop_up_objects to pop up objects.

SYNTAX

string **report_pop_up_object_options**
[-object_type *{pg_routing routing_guide blockage pin_guide pin_blockage cells}*]

ARGUMENTS

-object_type *{pg_routing routing_guide blockage pin_guide pin_blockage cells}*

Specifies the types of the objects whose options will be displayed. You can specify one or more of the supported object types by using the keywords: pg_routing, routing_guide, blockage, pin_guide, pin_blockage and cells. If *-object_type* is not specified, the settings for all object types are reported.

DESCRIPTION

This command reports options for command pop_up_objects.

EXAMPLES

The following example reports the options for *pg_routing* .

prompt> **report_pop_up_object_options -object_type pg_routing**

SEE ALSO

set_pop_up_object_options(2)
remove_pop_up_object_options(2)
pop_up_objects(2)

report_port

Displays port information within the design.

SYNTAX

string **report_ports** [-verbose]
[-design_rule]
[-drive]
[-input_delay]
[-output_delay]
[-nosplit]
[*port_list*]

list *port_list*

ARGUMENTS

-verbose

Indicates that the port report includes all port information. By default, only a summary section is displayed that lists all ports and their direction.

-design_rule

Reports only port design rule information, including maxCap, manLoad, and maxFanout.

-drive

Reports only drive resistance, input transition time, and driving cell information for only input and inout ports.

-input_delay

Reports only the port input delay information you set.

-output_delay

Reports only the port output delay information you set.

-nosplit

Prevents line splitting if column overflows. Most design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

port_list

Displays information on these ports in the current design. Each element in this list is either a collection of ports or a pattern

matching the port names.

DESCRIPTION

Displays information about ports in the current design. By default, the command produces a brief report including all ports in the design.

The input and output delay information lists the minimum rise, minimum fall, maximum rise, maximum fall delays, and the clock to which the delay is relative. If "(f)" follows the clock name, the delay is relative to the falling edge of the clock; otherwise the delay is relative to the rising edge. If "(l)" follows the clock name, input delay is considered to be coming from a level-sensitive latch, or output delay is considered to be going to a level-sensitive latch. By default, the delay is relative to a rising edge-triggered device.

Some information, such as whether the port is in an ideal network or not, is displayed after a timing update (as a result of manually executing an update_timing function or experiencing an implicit timing update as a result of executing other commands). This behavior is intended to help you to obtain information and statistics about ports without incurring the cost of a complete timing update.

EXAMPLES

This example shows the default port report.

```
prompt> report_ports

****************************************
Report : port
 ...

****************************************

Attributes:
   I - ideal network

                  Pin      Wire
Port         Dir    Cap      Cap     Attributes
-------------------------------------------------------
CLOCK          in     0.0000   0.0000
OPER           in     0.0000   0.0000
in0            in     0.0000   0.0000
in1            in     0.0000   0.0000
in2            in     0.0000   0.0000
out_1          out    0.0000   0.0000
out_2          out    0.0000   0.0000
out_3          out    0.0000   0.0000
out_4          out    0.0000   0.0000
p_in           in     0.0000   0.0000
p_out          out    0.0000   0.0000

1
```

SEE ALSO

report_design(2)
report_hierarchy(2)
report_net(2)
report_cell(2)
report_references(2)

report_port_buses

Displays port bus information within the design.

SYNTAX

string **report_port_buses**
[-verbose]
[-nosplit]
[*port_bus_list*]

Data Types

port_bus_list string or collection

ARGUMENTS

-verbose

Includes in the output the name of each member port. By default, a summary section is displayed that lists all port buses, their start bit, end bit, and bit order.

-nosplit

Prevents line splitting if the column overflows. Most design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

port_bus_list

Displays information only on the specified ports buses in the current design. *port_bus_list* contains either a collection of port buses or a pattern that matches the port buses names.

DESCRIPTION

This command displays information about port buses in the current design. By default, the command produces a brief report including all port buses in the current design, their start bit, end bit, and bit order.

EXAMPLES

This example outputs the default port bus report.

```
prompt> report_port_buses

****************************************
Report : report_port_buses
Design : ORCA
Version:
Date   :
****************************************
Name          Start  End  Bit order
----------------------------------------
pad           15     0    down
sd_DQ         15     0    down
pc_be         3      0    down
sd_A          9      0    down
sd_BWS        1      0    down
1
```

SEE ALSO

get_port_buses(2)
remove_port_buses(2)
report_design(2)
report_hierarchy(2)
report_nets(2)
report_cells(2)
report_references(2)

report_port_protection_diodes

Reports the port protection diodes in the current design.

SYNTAX

status **report_port_protection_diodes**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **report_port_protection_diodes** command reports the port protection diodes in the current design.

The reporting format is shown as below:

```
************************************************************
 Diode_Name        Location          Protected_Port
************************************************************

Number of ports with one diode             :
Number of ports with more than one diode   :
Number of ports with no diode              :
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the **report_port_protection_diodes** command.

```
prompt> report_port_protection_diodes
```

SEE ALSO

add_port_protection_diodes(2)
remove_cells(2)

report_ports

Displays port information within the design.

SYNTAX

string **report_ports** [-verbose]
[-design_rule]
[-drive]
[-input_delay]
[-output_delay]
[-nosplit]
[*port_list*]

list *port_list*

ARGUMENTS

-verbose

Indicates that the port report includes all port information. By default, only a summary section is displayed that lists all ports and their direction.

-design_rule

Reports only port design rule information, including maxCap, manLoad, and maxFanout.

-drive

Reports only drive resistance, input transition time, and driving cell information for only input and inout ports.

-input_delay

Reports only the port input delay information you set.

-output_delay

Reports only the port output delay information you set.

-nosplit

Prevents line splitting if column overflows. Most design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

port_list

Displays information on these ports in the current design. Each element in this list is either a collection of ports or a pattern

matching the port names.

DESCRIPTION

Displays information about ports in the current design. By default, the command produces a brief report including all ports in the design.

The input and output delay information lists the minimum rise, minimum fall, maximum rise, maximum fall delays, and the clock to which the delay is relative. If "(f)" follows the clock name, the delay is relative to the falling edge of the clock; otherwise the delay is relative to the rising edge. If "(l)" follows the clock name, input delay is considered to be coming from a level-sensitive latch, or output delay is considered to be going to a level-sensitive latch. By default, the delay is relative to a rising edge-triggered device.

Some information, such as whether the port is in an ideal network or not, is displayed after a timing update (as a result of manually executing an update_timing function or experiencing an implicit timing update as a result of executing other commands). This behavior is intended to help you to obtain information and statistics about ports without incurring the cost of a complete timing update.

EXAMPLES

This example shows the default port report.

```
prompt> report_ports

****************************************
Report : port
 ...

****************************************

Attributes:
   I - ideal network

                  Pin      Wire
Port         Dir    Cap      Cap     Attributes
-------------------------------------------------------
CLOCK          in     0.0000   0.0000
OPER           in     0.0000   0.0000
in0            in     0.0000   0.0000
in1            in     0.0000   0.0000
in2            in     0.0000   0.0000
out_1          out    0.0000   0.0000
out_2          out    0.0000   0.0000
out_3          out    0.0000   0.0000
out_4          out    0.0000   0.0000
p_in           in     0.0000   0.0000
p_out          out    0.0000   0.0000

1
```

SEE ALSO

report_design(2)
report_hierarchy(2)
report_net(2)
report_cell(2)
report_references(2)

report_power

Calculates and reports dynamic and static power for the design or instance.

SYNTAX

status **report_power**
[-net_power]
[-cell_power]
[objects *cell_or_net_list*]
[-hierarchy]
[-blocks]
[-early]
[-levels *level_value*]
[-leaf]
[-include_boundary_nets]
[-scenarios *scenario_list*]
[-modes *mode_list*]
[-corners *corner_list*]
[-significant_digits *digits*]
[-nosplit]
[-verbose]
[-force]

Data Types

cell_or_net_list list
level_value integer
scenario_list list
mode_list list
corner_list list
digits integer

ARGUMENTS

-net_power

Reports the power consumption of nets. Use the **-net_power** option alone, or use it with the **-cell_power** option. By default, only the summary power information for the design is reported when neither option is specified.

-cell_power

Reports the power consumption of cells. When you use the **-cell_power** option, some entries in the power report might not apply to certain cells. The column entry for such cells is annotated with N/A. Use the **-cell_power** option alone or with the **-net_power** option to report the cells and nets. By default, only the summary power information for the design is reported when neither option

is specified.

objects *cell_or_net_list*

Specifies a list of cells and nets to display with the **-net_power** or **-cell_power** options. With this option, only the cells and nets in the *cell_or_net_list* are listed in the power report. If both **-net_power** and **objects** options are specified, the *cell_or_net_list* must contain at least one net. Similarly, if both **-cell_power** and **objects** options are specified, the *cell_or_net_list* must contain at least one cell. If the **-net_power**, **-cell_power**, and **objects** options are specified together, the *cell_or_net_list* must contain at least one net and one cell.

-hierarchy

Specifies that the report is written in a hierarchical format, with power information reported on a block-by-block basis. Use the **-levels** option to limit the number of levels of hierarchy shown in the report. The switching, internal, and leakage power numbers are reported for each hierarchical block. The hierarchy is shown through indentations.

-blocks

Specifies that a physical-block oriented report should be written. The report will contain the per-physical-block power consumption for all power groups that are present in the block. Use the -levels option to indicate how many levels of physical hierarchy should be reported. The default number of levels is 0, indicating that no block-specific power should be reported. Only a accumulation of all sub-block power will be printed. A typical -levels value of 1 will print a power report for each first-level sub block and a power report for the top-level design. Power reporting with physical hierarchy supports abstract views of blocks, provided that the abstracts were generated with the *abstract.annotate_power* application option set to true.

-early

Reports power numbers for early timing. The default is to report for late timing.

-levels *level_value*

Specifies the number of levels of hierarchy to display in the hierarchical report. You can specify a positive integer greater than 1 with this option. Hierarchy levels deeper than those specified in this option are not shown in the hierarchical report. This option is only used for the hierarchical report and is ignored for all other types of reports.

-leaf

Specifies that the power report traverse the hierarchy and report objects at all lower levels (assumes a flat design hierarchy). The default behavior is to report objects at only the current level of hierarchy. For cell report, if the **-leaf** option is not specified, the power reported for a subdesign is the total power estimated for that subdesign, including all of its contents.

-include_boundary_nets

Specifies that the switching power of primary input nets must be counted when generating the power report; the default is to exclude boundary input nets.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Output of the command is generated separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command reports for the current scenario in the design.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering on modes will occur.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If

this option is not specified, no filtering on corners will occur.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 to 13. The default is 2.

-nosplit

Prevents line-splitting and facilitates writing scripts to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line starting in the correct column.

-verbose

Writes additional information to the report. By default, this option is enabled.

-force

Generates the power report for all the specified active/inactive scenarios irrespective of the leakage_power/dynamic_power flags of a scenario. Since timing data is not available for inactive scenarios, dynamic power numbers will not be accurate for -force option.

DESCRIPTION

The **report_power** command calculates and reports power for a design. The command uses the user-annotated switching activity to calculate the net switching power, cell internal power, and cell leakage power, and displays the calculated values in a power report. The **report_power** command requires switching activity information on all design nets, and uses a switching activity propagation mechanism to estimate switching activity information on non-annotated design objects.

The options enable you to specify cells and nets for reporting. The default operation is to display the summary of power values for only the current design. If user specified power groups are present then they are also reported in the summary along with the default power groups. Column 'Attrs' in summary report indicates whether the power group is user specified or default power group. If 'power.report_user_power_groups' is set as 'exclusive' then default and user power groups are reported in one table itself in the summary report. If the app option value is 'inclusive' then summary report has two tables, default power groups table followed by user power groups table.

Leakage power analysis is disabled if leakage_mode power app option is 'off' or leakage_power flag of a scenario is 'false'. Swcap power analysis is disabled if swcap_mode power app option is 'off' or dynamic_power flag of a scenario is 'false'. Internal power analysis is disabled if internal_mode power app option is 'off' or dynamic_power flag of a scenario is 'false'. The -force option can not override the leakage_mode / swcap_mode / internal_mode app options, but it can be used to override the leakage_power/dynamic_power flags of a scenario.

Multicorner-Multimode Support

EXAMPLES

The following example shows a summary report of the **report_power** command.

```
prompt> report_power
****************************************
Report : power
    -significant_digits 2
```

```
Design : ChipTop
Version:
Date   :
****************************************
Mode: default
Corner: default
Scenario: default
Voltage: 0.90
Temperature: 125.00

Voltage Unit        : V
Capacitance Unit    : pF
Time Unit           : ns
Temperature Unit    : C
Switching Power Unit : mW
Leakage Power Unit  : pW

  Cell Internal Power   = 0.00e+00  W  (0.0%)
  Net Switching Power   = 1.63e+00 mW  (100.0%)
Total Dynamic Power     = 1.63e-03  W  (100.0%)

Cell Leakage Power      = 2.00e+01 uW

  Attributes
  ----------
     u  -  User defined power group

Power Group       Internal     Switching      Leakage       Total
                    Power        Power         Power        Power     (   % )   Attrs
------------------------------------------------------------------------------------------
io_pad          0.00e+00       0.00e+00       0.00e+00      0.00e+00    ( 0.00%)
memory          0.00e+00       0.00e+00       0.00e+00      0.00e+00    ( 0.00%)
black_box       0.00e+00       0.00e+00       0.00e+00      0.00e+00    ( 0.00%)
clock_network   0.00e+00       5.37e-01       5.81e+04      5.37e-01    (32.62%)
register        0.00e+00       1.62e-01       5.05e+06      1.67e-01    (10.12%)
sequential      0.00e+00       0.00e+00       0.00e+00      0.00e+00    ( 0.00%)
combinational   0.00e+00       9.28e-01       1.49e+07      9.43e-01    (57.26%)
------------------------------------------------------------------------------------------
Total           0.00e+00 mW    1.63e+00 mW    2.00e+07 pW   1.65e+00 mW
```

The following example shows a cell and net power report for four objects in cell_or_net_list.

```
prompt> report_power -cell_power -net_power \
  {InstDecode/U18 InstDecode/U19 iso_cntl_2 iso_cntl} -corner [current_corner]
****************************************
Report : power
        -cell_power
        -net_power
        -significant_digits 2
Design : ChipTop
Version:
Date   :
****************************************
Mode: default
Corner: default
Scenario: default
Voltage: 0.90
```

```
Temperature: 125.00

Voltage Unit        : V
Capacitance Unit    : pF
Time Unit           : ns
Temperature Unit    : C
Switching Power Unit : mW
Leakage Power Unit  : pW



 Attributes
 ----------
    h  -  Hierarchical cell

            Cell            Driven Net     Total Dynamic             Cell
Cell Name     Internal Power  Switching Power  Power (% Internal/Dynamic)  Leakage Power  Attrs
-------------------------------------------------------------------------------------------------
InstDecode/U18      3.88e+04        2.79e+04         6.68e+04 ( 58.2%)      4.72e+02
InstDecode/U19      2.50e+05        2.33e+05         4.83e+05 ( 51.7%)      4.24e+02
-------------------------------------------------------------------------------------------------
Total (    2 cells)  2.89e+05 pW     2.61e+05 pW       5.50e+05 pW ( 52.5%)    8.96e+02 pW


 Attributes
 ----------
    a  -  Switching activity information annotated on net
    b  -  Switching activity information from abstract view
    s  -  Simulated switching activity information on net
    i  -  Implied switching activity information on net
    c  -  Constant switching activity information on net
    p  -  Switching activity information propagated on net
    t  -  STA switching activity information on net
    d  -  Default switching activity information on net
    e  -  Estimated switching activity information on net
    m  -  Net is driven by multiple pins

                 Total     Static        Toggle  Switching
Net Name        Vdd  Net Load  Probability    Rate    Power        Attrs
-----------------------------------------------------------------------------
iso_cntl_2  9.00e-01  0.00e+00     2.00e-01  2.33e-01   0.00e+00        d
iso_cntl    9.00e-01  1.86e-02     2.00e-01  2.33e-01   1.76e+06        d
-----------------------------------------------------------------------------
Total (    2 nets)                                 1.76e+06 pW
```

SEE ALSO

current_scenario(2)
get_scenarios(2)
report_scenarios(2)
set_scenario_status(2)
power.leakage_mode(3)
power.swcap_mode(3)

power.internal_mode(3)
power.propagation_effort(3)
power.clock_network_include_clock_gating_network(3)
power.clock_network_include_clock_sink_pin_power(3)
power.report_user_power_groups(3)
set_power_group(2)
reset_power_group(2)
report_power_groups(2)
get_power_group_objects(2)
get_power_group(2)

report_power_budget

Reports the power budget values for either the current design or a list of cells in the design.

SYNTAX

status **report_power_budget**
[-scenarios *scenario_list*]
[-significant_digits *digits*]
[-nosplit]
[-cell *cell_list*]

Data Types

scenario_list	list
digits	integer
cell_list	list

ARGUMENTS

-scenarios *scenario_list*

Indicates the scenario for which the budget values are reported. If no scenario is specified, the current_scenario is used.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 to 13. The default is 2.

-nosplit

Prevents line-splitting and facilitates writing scripts to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line starting in the correct column.

-cell *cell_list*

Specifies a list of cells for which the power budget values are reported.

DESCRIPTION

This command is used to report the budget values either for the design or for specific cells in the design.

If the command is called with the -cell option specified, then budget reports will only be issued for the instances specified in the cell

list.

For significant digits used in power budget report, the -significant_digits option can be used. By default, the number of significant digits used to display values is 2.

Multicorner-Multimode Support

EXAMPLES

The following example shows a summary report of the **report_power_budget** command.

```
prompt> report_power_budget
****************************************
Report : power_budget
Design : ChipTop
Version: P-2019.03-SP4-BETA
Date   : Tue Jul 23 07:38:25 2019
****************************************
Scenario: default

Object Name                    Object Type         Power Budget
-------------------------------------------------------------------------------

ChipTop                        Design              2.32e+08
1
```

The following example shows the budget report for the cell u1.

```
prompt> report_power_budget -cell u1
****************************************
Report : power_budget
Design : ChipTop
Version: P-2019.03-SP4-BETA
Date   : Tue Jul 23 07:41:18 2019
****************************************
Scenario: default

Object Name                    Object Type         Power Budget
-------------------------------------------------------------------------------

u1                             Cell                2.40e+06
1
```

SEE ALSO

set_power_budget(2)
get_power_budget(2)
reset_power_budget(2)
report_power(2)

report_power_calculation

Reports detailed power calculation for objects.

SYNTAX

status **report_power_calculation**
[-capacitance *pin_value_pair_list*]
[-transition *pin_value_pair_list*]
[-rise_transition *pin_value_pair_list*]
[-fall_transition *pin_value_pair_list*]
[-toggle_rate *pin_value_pair_list*]
[-probability *pin_value_pair_list*]
[-early]
[-rise]
[-fall]
[-scenarios *scenario_list*]
[-significant_digits *digits*]
[-verbose]
object_list

Data Types

pin_value_pair_list list
scenario_list list
digits integer
object_list string or collection

ARGUMENTS

-rise

Specifies that the internal power report should be limited to rise transitions only. If not specified, both rise and fall transitions are reported, which is equivalent to specifying both the **-rise** and **-fall** options.

-fall

Specifies that the internal power report should be limited to fall transitions only. If not specified, both rise and fall transitions are reported, which is equivalent to specifying both the **-rise** and **-fall** options.

-early

Reports power numbers for early timing. The default is to report for late timing.

-scenarios *scenario_list*

Specifies the scenarios to be considered for detailed power reporting. If this option is not specified, the scenario set currently in the design is considered.

-transition *pin_value_pair_list*

Specifies the total of rise and fall transitions that can be used to print the calculation of the virtual internal power of a module. Internally, half of this value is used for each of the rise and the fall transitions. It is specified as a list of {pin, value} pairs where the pin can either be a port in a module or a pin in an instance. The effect on internal power can be determined by varying the values specific to each port or pin of the module or the instance. Only one object can be specified in the **object_list** when this option is used and it has to be either a module or an instance. This option can not be specified with any of the **-rise_transition** and **-fall_transition** options.

-rise_transition *pin_value_pair_list*

Specifies the rise transition that can be used to print the calculation of the virtual internal power of a module. It is specified as a list of {pin, value} pairs where the pin can either be a port in a module or a pin in an instance. The effect on internal power can be determined by varying the values specific to each port or pin of the module or the instance. Only one object can be specified in the **object_list** when this option is used and it has to be either a module or an instance. This option can not be specified with the **-transition** option.

-fall_transition *pin_value_pair_list*

Specifies the fall transition that can be used to print the calculation of the virtual internal power of a module. It is specified as a list of {pin, value} pairs where the pin can either be a port in a module or a pin in an instance. The effect on internal power can be determined by varying the values specific to each port or pin of the module or the instance. Only one object can be specified in the **object_list** when this option is used and it has to be either a module or an instance. This option can not be specified with the **-transition** option.

-capacitance *pin_value_pair_list*

Specifies the total of rise and fall capacitance values that can be used to print the calculation of the virtual internal power of a module. Internally, half of this value is used for each of the rise and the fall capacitance values. It is specified as a list of {pin, value} pairs where the pin can either be a port in a module or a pin in an instance. The effect on internal power can be determined by varying the values specific to each port or pin of the module or the instance. Only one object can be specified in the **object_list** when this option is used and it has to be either a module or an instance.

-toggle_rate *pin_value_pair_list*

Specifies the total of rise and fall toggle rates that can be used to print the calculation of the virtual internal power of a module. Internally, half of this value is used for each of the rise and the fall toggle rates. It is specified as a list of {pin, value} pairs where the pin can either be a port in a module or a pin in an instance. The effect on internal power can be determined by varying the values specific to each port or pin of the module or the instance. Only one object can be specified in the **object_list** when this option is used and it has to be either a module or an instance.

-probability *pin_value_pair_list*

Specifies the probability value that can be used to print the calculation of the virtual internal power of a module. It is specified as a list of {pin, value} pairs where the pin can either be a port in a module or a pin in an instance. The effect on internal power can be determined by varying the values specific to each port or pin of the module or the instance. Only one object can be specified in the **object_list** when this option is used and it has to be either a module or an instance.

-significant_digits *digits*

Specifies the precision for floating point numbers to report. Allowed values are from 0 to 13. The default is 2.

-verbose

Writes detailed information to the report. Without this option, only the total power numbers for objects are shown.

object_list

Specifies the pin, net, instance or module for which to report the power calculation. If object is module or instance, with or without cap/tran/toggle/prob specified, internal/swcap/leakage power will be computed. If object is net, switching power will be computed else if object is pin, internal power will be computed.

DESCRIPTION

The **report_power_calculation** command provides detailed power calculation information for a specified pin, cell, module or net. You can use this information for debugging or verifying power data in a technology library. The application options - (power.internal_mode, power.swcap_mode, power.leakage_mode) control the output of this command. If the internal mode is off then the internal power is not reported at all. Similarly is the case with other application options. If all three application options are off then, a warning is issued just like it is done in the case of **report_power** command.

Multicorner-Multimode Support

EXAMPLES

The following example outputs a virtual internal power report for a module.

```
prompt> report_power_calculation stdcell_lib/MXI2X1M \
  -capacitance {{A 1.0} {B 2.0} {Y 6.0}} \
  -transition {{A 2.2} {S0 3.4} {Y 2.4}} \
  -toggle_rate {{A 11.0} {B 12.0} {Y 13.0}} \
  -probability {{A 0.8} {B 0.9} {Y 0.4}}

[INTERNAL POWER CALCULATION] (libcell: stdcell_lib/MXI2X1M)

Warning: Power table extrapolation (extrapolation mode) for port Y on
module stdcell_lib/MXI2X1M for parameter Cout. Lowest table
value = 0.000583, highest table value = 0.126985, value = 6.000000 (POW-046)
Warning: Power table extrapolation (extrapolation mode) for port Y on
module stdcell_lib/MXI2X1M for parameter Tinp. Lowest table
value = 0.047080, highest table value = 2.000000, value = 0.000000 (POW-046)

 Total Internal Power: 0.33171

---------------------------------------------------------------------
Total Internal Power for this libcell in scenario 'default' is 0.33171 mW
---------------------------------------------------------------------
```

The following example shows an internal power report for an instance when some of its parameters such as transition and probability have been modified.

```
prompt> report_power_calculation GPRs/U2199 \
  -capacitance {{A 1.0} {B 2.0} {Y 6.0}} \
  -transition {{A 2.2} {S0 3.4} {Y 2.4}} \
  -toggle_rate {{A 11.0} {B 12.0} {Y 13.0}} \
  -probability {{A 0.8} {B 0.9} {Y 0.4}}

[INTERNAL POWER CALCULATION] (cell: GPRs/U2199)
```

```
Warning: Power table extrapolation (extrapolation mode) for port Y
on cell GPRs/U2199 for parameter Cout. Lowest table value = 0.000583,
highest table value = 0.126985, value = 6.000000 (POW-046)

 Total Internal Power: 0.331765

---------------------------------------------------------------------
Total Internal Power for all cells in scenario 'default' is 0.331765 mW
---------------------------------------------------------------------
```

The following example shows the switching power section in the report for an instance that has been modified.

```
prompt> report_power_calculation Product[26] -verbose

[SWITCHING POWER CALCULATION] (net: Product[26])


Operating Environment:
   mode: default
   corner: default
   scenario: default
   voltage: 0.9
   temp: 125

Net Switching Power Calculation
net: Multiplier/Product[26]
driver: Multiplier/S_reg_reg_26_/Q

Switching power = 0.000169667
   net switching power = switching energy * net toggle rate

   Switching Energy Per Transition = 0.00509
     switching energy = 0.5 * capacitance * voltage ^ 2
     total net capacitance = 0.0125679
     voltage = 0.9
   Net Toggle Rate = 0.0333333 (default)

---------------------------------------------------------------------
Total Switching Power due to all nets in scenario 'default' is 0.000169667 mW
---------------------------------------------------------------------
```

SEE ALSO

report_power(2)

report_power_clock_scaling

Reports the clock frequency scaling values stored.

SYNTAX

status **report_power_clock_scaling**
[-scenarios *scenarios*]
[clock_list]

Data Types

scenarios list
clock_list list

ARGUMENTS

-scenarios *scenarios*

Specify a list of scenarios for which the scaling factor is to be retrieved, if found. If not specified, then scaling data is filtered for only the clock objs specified through clock_list option. If both scenarios and clocks are not specified then complete scaling data is printed.

clock_list

Specify a list of clocks in the design for which the scaling factor is to be retrieved, if found. If not specified, then scaling data is filtered for only the scenarios specified through -scenarios option. If both scenarios and clocks are not specified then complete scaling data is printed.

DESCRIPTION

Clock frequency used in Switching Activity Interchange Format (SAIF) files, from logic simulation for functional verification can differ from the clock frequency used in Synopsys Design Constraints (SDC) file for timing and power analysis. This command reports the scaling factors stored in the set_power_clock_scaling command.

The clock_objects option in this command is a list of clocks in the design that have the specified period or ratio values used in simulation for SAIF generation. The scenarios are the ones for which the scaling factors are to be retrieved and displayed. Note that if both the scenario and the clock options are not specified then the complete scaling data for all the scenarios is displayed.

When period value is specified for a clock, then the scaling ratio is calculated as the ratio of (period of the clock from set_power_clock_scaling cmd) / (period of the fastest clock of an object). This ratio is applied to the retrieved simulated activity of the object. If the ratio value is specified then it is directly applied as the scaling ratio.This ratio is applied to the retrieved simulated

activity of the object.

Multicorner-Multimode Support

EXAMPLES

The following examples describe the behaviour of the cmd.

```
prompt> set_power_clock_scaling {CLK CLK2 C_A} -period 2.6
prompt> set_power_clock_scaling -ratio 1
prompt> report_power_clock_scaling
------------------------------------------------------------
        Clock Scaling Data
------------------------------------------------------------
Scenario:default, Clock name:CLK, Period:2.600000
Scenario:default, Clock name:CLK2, Period:2.600000
Scenario:default, Clock name:C_A, Period:2.600000
------------------------------------------------------------
Scaling Ratios for Scenarios without Clocks
------------------------------------------------------------
Scenario:default, Ratio:1.000000
```

SEE ALSO

set_power_clock_scaling(2)
reset_power_clock_scaling(2)
get_power_clock_scaling(2)

report_power_derate

Reports the power derating factors for either the current design, a list of cells, or library cells in the current design. If no scenario is specified the current scenario is used.

SYNTAX

status **report_power_derate**
[-scenarios *scenario_list*]
[-include_inherited]
[-significant_digits *digits*]
[-nosplit]
[*object_list*]

Data Types

scenario_list	list
digits	integer
object_list	list

ARGUMENTS

-scenarios *scenario_list*

Indicates the scenario for which the derate_value is applied. If no scenario is specified the current_scenario is used.

-include_inherited

Indicates that the power derating factors reported on each object should also include those set by other objects in the design.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 to 13. The default is 2.

-nosplit

Prevents line-splitting and facilitates writing scripts to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line starting in the correct column.

object_list

Specifies current design or a list of cells or library cells to which the specified power derating factor is applied. Note groups and object_list are mutually exclusive option.

DESCRIPTION

This command is used to report the derate factors either on the design, specific cells or library cells of the design. If this command is called without any option, a separate report will be invoked for the power derating factors stored on the current design and also for each cell or library cell that has power derating factors set on it.

If this command is called with option -include_inherited each row containing an inherited derating factor will contain a column entry indicating the object from which it was inherited. If the command is called with the object_list specified then derate reports will only be issued for the instances specified in the object list.

For significant digits used in power derate report, the -significant_digits option can be used. By default, the number of significant digits used to display values is 2.

Multicorner-Multimode Support

EXAMPLES

The following example shows a summary report of the **report_power_derate** command.

```
prompt> report_power_derate
****************************************
Report : power_derate
Design : Top
Version:
Date   :
****************************************
Scenario: default



 Attributes
 ----------
    h  -  Hierarchical cell
Object Name  Object Type    Leakage    Switching    Internal    Attrs
--------------------------------------------------------------------------------

Top             design       N/A         N/A     1.50e+00

M1               cell        N/A      3.00e+00        N/A       h

M1/U1            cell     2.50e+00         N/A         N/A

Lib1/Mod1      lib cell       N/A      3.50e+00        N/A

Lib1/Mod2      lib cell     2.00e+00         N/A         N/A
```

The following example shows a cell and its inherited derate report

```
prompt> report_power_derate -include_inherited { M1/U1 }
****************************************
Report : power_derate
Design : Top
Version:
```

```
Date   :
****************************************
Scenario: default


 Attributes
 ----------
    h  -  Hierarchical cell
Object Name  Object Type    Leakage    Switching    Internal    Attrs
--------------------------------------------------------------------------------

M1/U1            cell   2.50e+00     3.00e+00     1.50e+00

Inherited                  N/A        M1      Top
```

SEE ALSO

set_power_derate(2)
get_power_derate(2)
reset_power_derate(2)

report_power_domain

Reports information about the specified power domains.

SYNTAX

status **report_power_domains**
[-nosplit]
[*power_domains*]

Data Types

power_domains list

ARGUMENTS

-nosplit

Does not split lines if column overflows.

power_domains

Specifies the power domains to be reported. The command fails if it does not find any of the specified power domains. If this option is not specified the command reports all power domains in the current scope.

DESCRIPTION

The **report_power_domains** command reports detailed information about existing power domains. If you specify the power domains, only those power domains are reported; otherwise all power domains in the current scope are reported.

Until commit_upf command is executed and power intent is finalized, power domains of subblocks are not propagated and will not be included in the report.

The following information is reported for each power domain:

- The full name of the power domain
- The current scope
- The elements of the power domain
- The voltage area of the power domain
- The supply nets available for use in the power domain

- The supply sets available for use in the power domain
- The default supply nets of the power domain and their voltages if set using the **set_voltage** command
- The power switches of the power domain
- The isolation strategies of the power domain
- The level shifter strategies of the power domain
- The repeater strategies of the power domain
- The retention strategies of the power domain

EXAMPLES

The following example reports all the power domains in the current scope:

```
prompt> report_power_domains
****************************************
Report : Power Domain
Design : top
Corner : default
 ...

****************************************
--------------------------------------------------------------------------------
Power Domain          : TOP
Current Scope         : top (top level)
Elements              : top
Voltage Area          : DEFAULT_VA
Available Supply Nets : VDD, VSS, VDDMA
Available Supply Sets : SS2

Default Supplies         -- Power --              -- Ground --
  Primary             : VDD                    VSS
  Isolation           : --                     --
  Retention           : --                     --

Power Switches        :
Isolation Strategy    : snps_no_iso
  Applies To          : both
  No Isolation        : True
  Elements            : iso_en
  Mapped Cells        : N/A

Level Shifter Strategy :
Repeater Strategy     : snps_rptr
  Supply Set          : SS2
  Applies To          : outputs
  Elements            : N/A
  Excluded Elements   : N/A

Retention Strategy    :
```

```
--------------------------------------------------------------------------------
1

The following example reports all the power domains in
the current design:

prompt> report_power_domains [get_power_domains -hierarchical]
****************************************
Report : Power Domain
Design : top
Corner : default
 ...

****************************************
--------------------------------------------------------------------------------
Power Domain          : TOP
Current Scope         : top (top level)
Elements              : top
Voltage Area          : DEFAULT_VA
Available Supply Nets : VDD, VSS, VDDMA
Available Supply Sets :

Default Supplies        -- Power --              -- Ground --
  Primary             : VDD                      VSS
  Isolation           : --                       --
  Retention           : --                       --

Power Switches        :
Isolation Strategy    : snps_no_iso
  Applies To          : both
  No Isolation        : True
  Elements            : iso_en
  Mapped Cells        : N/A

Level Shifter Strategy :
Repeater Strategy      :
Retention Strategy     :
--------------------------------------------------------------------------------
Power Domain          : InstLS_B/PD
Current Scope         : top (top level)
Elements              : InstLS_B
Voltage Area          : N/A
Available Supply Nets : InstLS_B/VDDM, InstLS_B/VDD, InstLS_B/VSS
Available Supply Sets :

Default Supplies        -- Power --              -- Ground --
  Primary             : InstLS_B/VDD             InstLS_B/VSS
  Isolation           : --                       --
  Retention           : --                       --

Power Switches        :
Isolation Strategy    : ls_iso
  Power               : InstLS_B/VDD
  Ground              : InstLS_B/VSS
  Clamp               : 0
  Signal              : InstLS_B/iso
```

```
    Sense            : low
    Location         : self
    Applies To       : inputs
    Elements         : N/A
    Mapped Cells     : N/A

  Level Shifter Strategy :
  Repeater Strategy      :
  Retention Strategy     :
  ----------------------------------------------------------------------------
  1
```

SEE ALSO

create_power_domain(2)
get_power_domains(2)

report_power_domains

Reports information about the specified power domains.

SYNTAX

status **report_power_domains**
[-nosplit]
[*power_domains*]

Data Types

power_domains list

ARGUMENTS

-nosplit

Does not split lines if column overflows.

power_domains

Specifies the power domains to be reported. The command fails if it does not find any of the specified power domains. If this option is not specified the command reports all power domains in the current scope.

DESCRIPTION

The **report_power_domains** command reports detailed information about existing power domains. If you specify the power domains, only those power domains are reported; otherwise all power domains in the current scope are reported.

Until commit_upf command is executed and power intent is finalized, power domains of subblocks are not propagated and will not be included in the report.

The following information is reported for each power domain:

- The full name of the power domain
- The current scope
- The elements of the power domain
- The voltage area of the power domain
- The supply nets available for use in the power domain

- The supply sets available for use in the power domain
- The default supply nets of the power domain and their voltages if set using the **set_voltage** command
- The power switches of the power domain
- The isolation strategies of the power domain
- The level shifter strategies of the power domain
- The repeater strategies of the power domain
- The retention strategies of the power domain

EXAMPLES

The following example reports all the power domains in the current scope:

```
prompt> report_power_domains
****************************************
Report : Power Domain
Design : top
Corner : default
 ...

****************************************
--------------------------------------------------------------------------------
Power Domain          : TOP
Current Scope         : top (top level)
Elements              : top
Voltage Area          : DEFAULT_VA
Available Supply Nets : VDD, VSS, VDDMA
Available Supply Sets : SS2

Default Supplies        -- Power --              -- Ground --
  Primary             : VDD                      VSS
  Isolation           : --                       --
  Retention           : --                       --

Power Switches        :
Isolation Strategy    : snps_no_iso
  Applies To          : both
  No Isolation        : True
  Elements            : iso_en
  Mapped Cells        : N/A

Level Shifter Strategy :
Repeater Strategy     : snps_rptr
  Supply Set          : SS2
  Applies To          : outputs
  Elements            : N/A
  Excluded Elements   : N/A

Retention Strategy    :
```

```
--------------------------------------------------------------------------------
1

The following example reports all the power domains in
the current design:

prompt> report_power_domains [get_power_domains -hierarchical]
****************************************
Report : Power Domain
Design : top
Corner : default
 ...

****************************************
--------------------------------------------------------------------------------
Power Domain          : TOP
Current Scope         : top (top level)
Elements              : top
Voltage Area          : DEFAULT_VA
Available Supply Nets : VDD, VSS, VDDMA
Available Supply Sets :

Default Supplies        -- Power --              -- Ground --
  Primary             : VDD                      VSS
  Isolation           : --                       --
  Retention           : --                       --

Power Switches        :
Isolation Strategy    : snps_no_iso
  Applies To          : both
  No Isolation        : True
  Elements            : iso_en
  Mapped Cells        : N/A

Level Shifter Strategy :
Repeater Strategy     :
Retention Strategy    :
--------------------------------------------------------------------------------
Power Domain          : InstLS_B/PD
Current Scope         : top (top level)
Elements              : InstLS_B
Voltage Area          : N/A
Available Supply Nets : InstLS_B/VDDM, InstLS_B/VDD, InstLS_B/VSS
Available Supply Sets :

Default Supplies        -- Power --              -- Ground --
  Primary             : InstLS_B/VDD             InstLS_B/VSS
  Isolation           : --                       --
  Retention           : --                       --

Power Switches        :
Isolation Strategy    : ls_iso
  Power               : InstLS_B/VDD
  Ground              : InstLS_B/VSS
  Clamp               : 0
  Signal              : InstLS_B/iso
```

```
    Sense            : low
    Location         : self
    Applies To       : inputs
    Elements         : N/A
    Mapped Cells     : N/A

  Level Shifter Strategy :
  Repeater Strategy      :
  Retention Strategy     :
  ----------------------------------------------------------------------------
  1
```

SEE ALSO

create_power_domain(2)
get_power_domains(2)

report_power_groups

Reports existing power groups and number of cells attached to that power group.

SYNTAX

status **report_power_groups**
[*group_names*]

Data Types

group_names list

ARGUMENTS

group_names

Specifies the power group names that needs to be reported. If this option is not present, all the power groups (user specified and default power groups) are reported.

DESCRIPTION

The report_power_groups command reports all the existing power groups(user and default power groups) and number of cells present in each of these power groups. The report depends on the value of app option 'power.report_user_power_groups'

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example report power groups macro1 and memory.

prompt> **report_power_groups {macro1 memory}**

SEE ALSO

set_power_group(2)
reset_power_group(2)
get_power_group(2)
get_power_group_objects(2)
report_power(2)
power.report_user_power_groups(3)

report_power_io_constraints

Reports power constraints set on specified I/O guides.

SYNTAX

int **report_power_io_constraints**
[-io_guide_list *io_guides*]
[-significant_digits *digits*]

Data Types

io_guides list
digits int

ARGUMENTS

-io_guide_list

Specifies the names or collection of the target I/O guides. If this option is not used, all I/O guides are selected.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13; the default is determined by the **shell.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default.

DESCRIPTION

This command reports power constraints set on I/O guides. There are four power constraint categories for each type of power pads: ratio, share, spacing, and offset. If there is no constraint on any category, nothing is reported. If some categories, but not all categories, have constraints, the unconstrained categories are reported with the string "no constraint".

EXAMPLES

The following example reports power constraints on an I/O guide named "north".

prompt> **report_power_io_constraints -io_guide_list north**

The following example reports power constraints on all I/O guides.

```
prompt> report_power_io_constraints
```

SEE ALSO

place_io(2)
set_signal_io_constraints(2)
set_power_io_constraints(2)
remove_power_io_constraints(2)

report_power_model

Report information of power models in the design.

SYNTAX

status **report_power_model**
[-csv]
[-example]
[-html]
[-output *output_file*]
[-verbose]

ARGUMENT

-csv

Print report in CSV (Comma-Separated Value) format.

-example

Display a list of typical example usages of this command.

-html

Print report in HTML format.

-output <*output_file*>

Print report into <*output_file*>.

-verbose

Report detailed information about power models in the design.

DESCRIPTION

This command reports power model information. It includes information about power domains created in the **define_power_model** command and applied in the **apply_power_model** command.

In verbose mode, it also displays detailed information about the options used in the **define_power_model** and **apply_power_model** commands.

EXAMPLES

The following example reports all power models found in the design:

```
prompt> report_power_model
****************************************
Report : report_power_model
Design : top
Version: O-2018.06-SP2
Date   : Thu Jul 26 13:54:58 2018
****************************************

Column Header
   Elmt - Num of cells in apply_power_model -elements
   Smap - Num of mappings in apply_power_model -supply_map
   Pmap - Num of mappings in apply_power_model -port_map
   Cell - Num of cells applied with power model

 Power Model        Defined In         Applied In         Elmt SMap PMap Cell
 ------------------- ------------------- ------------------- ---- ---- ---- ----
 macro_lib_cell_1    User Library       Top Scope          0    0    0    5
 power_model_1       Top Scope          Top Scope          1    2    0    1
 power_model_2       MID_0              MID_0              2    0    3    2
 power_model_3       MID_0/BOT_0        MID_0/BOT_0        3    0    0    3
```

SEE ALSO

apply_power_model(2)
define_power_model(2)

report_power_scopes

Reports information about the power scopes of the design.

SYNTAX

status **report_power_scopes**
[-nosplit]

ARGUMENTS

-nosplit

Does not split lines if column overflows.

DESCRIPTION

The **report_power_scopes** command enables you to get detailed information about existing power scopes in the current design.

Until commit_upf command is executed and power intent is finalized, power intent of subblocks are not propagated and will not be included in the report.

The report includes the name of the scope and all the power domains, supply ports, supply nets, supply sets and power state tables that are defined in each scope.

EXAMPLES

The following example reports all the power scopes in the current design:

```
prompt> report_power_scopes
****************************************
Report : Power Scope
 ...

****************************************
--------------------------------------------------
Scope           : top (top level)
Power Domains   : pd
Supply Ports    : vdd, vdd1
```

```
    Supply Nets       : vdd, vdd1
    Supply Sets       : vddset, vddset1
    Power State Tables : pst, pst1
    ---------------------------------------------------
    Scope             : U11
    Power Domains     : pd3
    Supply Ports      : vdd_port, vdd_port1
    Supply Nets       : vdd, vdd1
    Supply Sets       : vddset, vddset1
    Power State Tables :
    ---------------------------------------------------
```

SEE ALSO

report_power_domains(2)
report_supply_ports(2)
report_supply_nets(2)
report_pst(2)

report_power_switch_patterns

Reports power switch patterns.

SYNTAX

status **report_power_switch_patterns**
[-nosplit]
[*pattern_name*]

Data Types

pattern_name string

ARGUMENTS

-nosplit

Does not split lines if column overflows.

pattern_name

Specifies the power switch pattern name. By default, the command reports all the defined power switch patterns. Report includes the HEAD pin and TAIL pins of the patterns specified. Please note, in case of HFN connections only HEAD pins will be reported.

DESCRIPTION

This command reports power switch patterns.

EXAMPLES

The following example reports the power switch patterns in the design.

```
prompt> report_power_switch_patterns
Pattern             :
  Head
    Tail
PSW_PATTERN_INST_pattern2_0 :
  InstDecode/headerfooter_snps_INST__inst_sw_snps_HEAD64DM_R0_C0_0/SLEEP
```

InstDecode/headerfooter_snps_INST__inst_sw_snps_HEAD32DM_R3_C0_3/SLEEPOUT

SEE ALSO

create_power_switch_ring(2)
create_power_switch_array(2)
set_power_switch_placement_pattern(2)
get_power_switch_patterns(2)
connect_power_switch(2)

report_power_switch_placement_patterns

Reports power switch placement patterns.

SYNTAX

status **report_power_switch_placement_patterns**
[*pattern_name*]

Data Types

pattern_name string

ARGUMENTS

pattern_name

Specifies the power switch placement pattern name. By default, the command reports the power switch placement patterns for all defined pattern names.

DESCRIPTION

This command reports power switch placement patterns.

EXAMPLES

The following example reports the power switch placement patterns in the design.

prompt> **report_power_switch_placement_patterns**

SEE ALSO

create_power_switch_ring(2)
create_power_switch_array(2)
set_power_switch_placement_pattern(2)

report_pr_rules

Reports information about the cell row spacing rules defined in the specified technology object.

SYNTAX

status **report_pr_rules**
[-tech *tech_object*]
[-library *library*]
[-nosplit]
[-all]
[*patterns*]

Data Types

tech_object collection
library collection
patterns collection

ARGUMENTS

-tech *tech_object*

Specifies the technology object from which to report the cell row spacing rules.

The **-tech** and **-library** options are mutually exclusive; you can specify only one. If you do not specify either option, the rules are reported from the technology object of the current library.

-library *library*

Specifies the library from which to report the cell row spacing rules. The command reports the rules from the technology object associated with the specified library. In an implementation tool, the library is a design library. In the library manager, the library is a cell library. You can specify the library using the library's name or a collection that contains the library.

The **-tech** and **-library** options are mutually exclusive; you can specify only one. If you do not specify either option, the rules are reported from the technology object of the current library.

-nosplit

Prevents line splitting if the column overflows. Most information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

This option prevents line-splitting and facilitates writing software to extract information from the report output.

-all

Reports all cell row spacing rules.

The **-all** option and *patterns* argument are mutually exclusive; you must specify one.

patterns

Reports the cell row spacing rules that match the specified patterns. You can specify a list of patterns or a collection of cell row spacing rules.

The **-all** option and *patterns* argument are mutually exclusive; you must specify one.

DESCRIPTION

This command reports information about the cell row spacing rules in the specified technology object.

EXAMPLES

The following example reports information about all the cell row spacing rules in the technology object of the current library.

```
prompt> report_pr_rules -all
****************************************
Report : report_pr_rules
****************************************
Name            Property Name          Value
-------------------------------------------------
PR_RULE_0       row_spacing_top_top    0
                row_spacing_top_bot    1.03
                row_spacing_bot_bot    0
                abut_table_top_top     1
                abut_table_top_bot     0
                abut_table_bot_bot     1
1
```

The following example reports information about the cell row spacing rule named PR_RULE_0.

```
prompt> report_pr_rules PR_RULE_0
****************************************
Report : report_pr_rules
****************************************
Name            Property Name          Value
-------------------------------------------------
PR_RULE_0       row_spacing_top_top    0
                row_spacing_top_bot    1.03
                row_spacing_bot_bot    0
                abut_table_top_top     1
                abut_table_top_bot     0
                abut_table_bot_bot     1
1
```

SEE ALSO

get_pr_rules(2)
remove_pr_rules(2)
create_pr_rule(2)
report_hierarchy(2)
report_nets(2)
report_cells(2)
report_references(2)

report_programmable_spare_cell_mapping_rule

This command report rules for programmable spare cells mapping.

SYNTAX

status **report_programmable_spare_cell_mapping_rule**
-psc_type_id *id_list* | -all

Data Types

id_list list

ARGUMENTS

-psc_type_id *id_list*

Specify the list of psc_type_id of which the mapping rules will be reported. This option is mutually exclusive with option *-all*.

-all *id_list*

Report all mapping rules for all psc types. This option is mutually exclusive with option *-psc_type_id*.

DESCRIPTION

This command report rules for programmable spare cells mapping. Users can specify the list of psc_type_id to report mapping rules or report all mapping rules for all types.

The report summary is as the following.

```
****************************************
Report : PSC Mapping Rule Report
Design : r4000
Version: M-2016.12-SP2-BETA
Date   : Tue Jan 19 22:19:13 2017
****************************************
   PSC Type 1:
      Not Overlap Layers:    M1
      Compatible PSC Types:  2

   PSC Type 2:
```

```
    Partial Overlap Layers: M1
    Partial Overlap Side:   Right
    Compatible PSC Types:   1

  PSC Type 3:
    Partial Overlap Layers: M1
    Partial Overlap Side:   Right
    Split PSC Types:        2
    Merge PSC Types:        1, 2
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example report the rule for psc_type_id 1 and 2.

```
prompt> report_programmable_spare_cell_mapping_rule -psc_type_id {1 2}\
```

The following example report all rules.

```
prompt> report_programmable_spare_cell_mapping_rule -all\
```

SEE ALSO

set_programmable_spare_cell_mapping_rule(2)
remove_programmable_spare_cell_mapping_rule(2)
place_freeze_silicon(2)
map_freeze_silicon(2)

report_pst

Reports the power state table(s) in the current design.

SYNTAX

status **report_pst**
[-derived]
[-reconcile]
[-psts *pst_list*]
[-supplies *supply_list*]
[-voltage_type *voltage_type*]
[-nosplit]

Data Types

pst_list list
supply_list list
voltage_type none|nominal|all

ARGUMENT

-derived

Report the derived PST of the design.

-reconcile

Report the voltage reconciliation settings of the design. Reconciliation setting can be done by set_design_attribute of upf_reconcile_boundary attribute and set_variation command. The report is done after resolving supply variations between top level design and block level designs.

-psts *pst_list*

Specifies a list of user-defined power state tables to be included in the report. By default, all used-defined power state tables in the current scope and child scopes are included. This option is mutually exclusive with the **-derived** option. When referencing any PSTs that are not in the top scope, the scope of the PST must be given. For example, if a PST name 'pst' is defined at scope 'u0', it must be referenced as 'u0/pst'.

-supplies *supply_list*

Specifies a list of supply nets, supply ports, or supply set functions to be included in the report. The order of supplies in this list also determines the order they are shown in the report. For user-defined power state table, the report shows the specified supply name followed by the supply name defined in the table if they are different but equivalent. By default, all supplies in the current design are included.

-voltage_type *voltage_type*

Specifies the voltage type to be included in the report. If the value is 'nominal', only the nominal voltage value of supply state(s) will be included. If the value is 'all', all defined voltage values of supply state(s) will be included. By default, the value is 'none' and voltage values are not included in the report.

-nosplit

Does not split lines if column overflows.

DESCRIPTION

This command reports the power state table(s) of the current design. If **-derived** option is not specified, the report will include user-defined power state tables of the design. If **-derived** option is specified, the derived power state table which combines all user-defined power state tables contained in the current design or in its logical hierarchy will be shown.

Until commit_upf command is executed and power intent is finalized, power state tables of subblocks are not propagated and will not be included in the report.

If the power state tables has not been created, the power state table is full, meaning that any power state is allowed. Once the power state table is created, the table becomes empty and new power states could be created and added to the table using the **add_pst_state** command.

A "*" in place of a state name in the report indicates this is a "don't care" for that supply.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

SEE ALSO

add_power_state(2)
add_port_state(2)
add_pst_state(2)
create_pst(2)
set_variation(2)

report_pt_options

Report configuration options specified by the **set_pt_options** command

SYNTAX

status **report_pt_options**

ARGUMENTS

None.

DESCRIPTION

The **report_pt_options** command reports all the options specified by the **set_pt_options** command.

EXAMPLES

The example below reports all the options.

prompt> **report_pt_options**

SEE ALSO

set_pt_options(2)
eco_opt(2)
check_pt_qor(2)

report_push_down_object_options

Reports the current option settings for the **push_down_objects** command. The options are set with the **set_push_down_object_options** command.

SYNTAX

string **report_push_down_object_options**
[-object_type {*types*}

Data Types

types list

ARGUMENTS

-object_type {*types*}

Specifies the types of objects for which to display current option settings. Specify one or more of the supported object types by using the following keywords: **pg_routing**, **signal_routing**, **routing_guide**, **routing_corridor**, **blockage**, **pin_guide**, **pin_blockage**, **cells**, and **charging_station**.

If **-object_type** is not specified, the command reports the settings for all object types.

DESCRIPTION

This command reports option settings for the **push_down_objects** command. Options are set with the **set_push_down_object_options** command.

EXAMPLES

The following example reports the options for *pg_routing*.

prompt> **report_push_down_object_options -object_type pg_routing**

The following example reports the options for *routing_corridor*.

prompt> **report_push_down_object_options -object_type routing_corridor**

SEE ALSO

push_down_objects(2)
remove_push_down_object_options(2)
set_push_down_object_options(2)

report_pvt

Reports per-corner PVT information, including mismatches between specified and effective PVT values.

SYNTAX

string **report_pvt**
  [-object_list *objects*]
  [-mismatched]
  [-nosplit]
  [-significant_digits *digits*]
  [*corner_list*]

list *corner_list*
list *objects*

ARGUMENTS

-object_list *objects*

Specifies a list of cell instances or ports to report PVT information for.

-mismatched

If given with the **-object_list** option, this will cause only the objects with specified vs. effective PVT mismatches to be reported.

-nosplit

Indicates that lines are not to be split when column fields overflow. This is most useful when comparing previous script files with the UNIX diff command, or for post-processing the script.

-significant_digits *digits*

Specifies the number of digits after the decimal point to be displayed for values in the report. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option. Use this option if you want to override the default.

corner_list

Lists corners to report. If you do not specify this option, all corners in the current design are reported.

DESCRIPTION

Displays information about the PVT conditions used by the cells and ports of each corner of the design. The PVT parameters include the specified and effective values of the process label, the process number, the temperature, and the voltages of each power rail, for both early and late analysis types. The actual early and late pane numbers are also reported.

If no options are given, *report_pvt* will report all mismatched PVT conditions, and the number of cells and ports that use each one.

If the **-object_list** option is given, the command will report the specified and effective PVT parameters used by each cell or port in the list. If the **-mismatched** option is also given, the report will be filtered to include only the objects with mismatched specified vs. effective PVT parameters.

Multicorner-Multimode Support

This command works on all corners.

SEE ALSO

report_corner(2)
set_process_label(2)
set_process_number(2)
set_voltage(2)
set_temperature(2)

report_qor

Displays QoR information and statistics for the current design.

SYNTAX

status **report_qor**
[-include *content_list*]
[-summary]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-significant_digits *digits*]
[-pba_mode none | path | exhaustive]
[-nosplit]

Data Types

mode_list collection
corner_list collection
scenario_list collection
digits integer

ARGUMENTS

-include *content_list*

Specifies the types of information to be reported. Valid list entries are: design_stats, electrical_drc, hold, and setup.

-summary

Gives a summary of the design costs.

-modes *mode_list*

Each active scenario of the modes given by the **-modes** option and the corners specified by the **-corners** option will be reported. If the **-corners** option is not given with this option, all corners will be used. If neither a mode, corner, nor scenario list is given, the current scenario (if any) will be reported. If the **-scenarios** option is given, this option is not allowed.

-corners *corner_list*

Each active scenario of the modes given by the **-modes** option and the corners specified by the **-corners** option will be reported. If the **-modes** option is not given with this option, all modes will be used. If neither a mode, corner, nor scenario list is given, the current scenario (if any) will be reported. If the **-scenarios** option is given, this option is not allowed.

-scenarios *scenario_list*

Each active scenario given by the **-scenarios** option will be reported. If this option is given, the **-mode** and **-corner** options are not allowed.

-significant_digits ***digits***

Specifies the number of digits to the right of the decimal point that are reported. Allowed values are from 0 through 13. The default value is 2 for all sections except the Area section, which has a default value of 6. Using this option overrides the value set by the **shell.common.report_default_significant_digits** application option.

-pba_mode none | **path** | **exhaustive**

Controls path-based analysis with the following modes:

- **none** (the default) - Path-based analysis is not applied.
- **path** - Path-based analysis is applied over the paths that gave the worst graph-based analysis violation. There is no guarantee that the reported path-based analysis slack over every endpoint is the worst one.
- **exhaustive** - An exhaustive path-based analysis path search algorithm is applied to determine the worst path-based analysis slack violations.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds its column's width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

This command reports Quality of Results information about the design. This includes timing information, cell count details, and statistics such as combinational, noncombinational, and total area.

Timing-related information is reported separately for each scenario specified by the **-mode**, **-corner**, and **-scenario** options.

Under the Cell Count section, the Leaf Cell Count report includes all leaf cells that are not constant cells. Constant cells are omitted in this count. The Combinational Cell Count and Sequential Cell Count only include leaf cells.

Under Area section, Cell Area (netlist) ignores physical-only cells but Cell Area (netlist and physical only) includes them for area calculation. Later one is identical to Total physical cell area reported by "report_design -physical" command.

If you specify the **-pba_mode exhaustive** option, path-based analysis is used to calculate the slack values. This option requires that a path search be performed to ensure that the returned slack values are truly the worst. The additional runtime required for this option depends on the amount of timing improvement resulting from path-based analysis. As the timing improvement from path-based analysis increases, the runtime for the path search increases.

Multicorner-Multimode Support

By default, this command works on all active scenarios. To specify different scenarios, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following is an example of a QoR report:

```
prompt> report_qor

****************************************
Report : qor
Design : top
****************************************


Scenario         'func::090V_125C'
Timing Path Group  'CLK1'
-----------------------------------------
Levels of Logic:                17
Critical Path Length:            2.66
Critical Path Slack:            -0.69
Critical Path Clk Period:        2.20
Total Negative Slack:         -597.07
No. of Violating Paths:          2640
-----------------------------------------

Scenario         'func::090V_125C'
Timing Path Group  'CLK2'
-----------------------------------------
Levels of Logic:                12
Critical Path Length:            2.88
Critical Path Slack:            -2.68
Critical Path Clk Period:        2.20
Total Negative Slack:         -483.99
No. of Violating Paths:           497
-----------------------------------------


Cell Count
-----------------------------------------
Hierarchical Cell Count:           25
Leaf Cell Count:                30852
Buf/Inv Cell Count:              6255
CT Buf/Inv Cell Count:              0
-----------------------------------------


Area
-----------------------------------------
Combinational Area:         175600.52
Noncombinational Area:      186276.49
Net Area:                           0
Net XLength:                        0
Net YLength:                        0
-----------------------------------------
Cell Area (netlist):              361877.02
Cell Area (netlist and physical only): 523152.83
Net Length:                         0


Design Rules
```

```
----------------------------------------
Total Number of Nets:          31577
Nets with Violations:          7624
Max Trans Violations:          196
----------------------------------------

1
```

SEE ALSO

shell.common.report_default_significant_digits(3)

report_qor_snapshot

Reports an existing QoR snapshot.

SYNTAX

void **report_qor_snapshot**
[-name *name*]
[-directory *snapshot_directories*]
[-display]
[-save_as *report_name*]

Data Types

name string
snapshot_directories list
report_name string

ARGUMENTS

-name *name*

Specifies the name of the QoR snapshot to be reported. If you do not specify this option, the tool reports all the QoR snapshots in the snapshot directory.

-directory *snapshot_directories*

Specifies the location of the snapshots. If you do not specify this option, the tool looks for the QoR snapshot in the directory specified by the application option **time.snapshot_storage_location**.

-display

Displays the QoR report in the interactive browser.

-save_as *report_name*

Specifies the name of the report. The report is saved under this name; you can then can access the report at a later time using this name.

DESCRIPTION

This command reports the specified QoR snapshots that are located in the specified directory. The tool issues an error if no snapshot exists.

The output of the report is stored at the location specified by the application option **time.snapshot_storage_location**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

In the following example, the **report_qor_snapshot** command reports the QoR snapshot named *preroute* which is saved in the snapshot directory.

```
prompt> report_qor_snapshot -name preroute
```

SEE ALSO

create_qor_snapshot(2)
remove_qor_snapshot(2)
query_qor_snapshot(2)
time.snapshot_storage_location(3)

report_rail_minimum_path

Reports resistance and voltage drop values per layer in minimum resistance path based on the current rail result.

SYNTAX

status **report_rail_minimum_path**
-net *supply_net*
-cell *cell*
-error_cell
[-all]
[-verbose]

Data Types

cell_collection *cell*
supply_net_collection *supply_net*

ARGUMENTS

-cell *cell_object*

Specifies the cell object to be reported.

-net *supply_net_object*

Specifies the supply net object to be reported.

error_cell

Writes the minimum resistance path to the error view named rail_min_path_resistance.<cell_name>.<net_name>.err

all

Reports all pin shapes.

verbose

Shows the detailed report.

DESCRIPTION

The command reports the minimum resistance path based on the current rail result. The minimum resistance path is the shortest

distance from each PG pin which is connected to the given supply net to tap.

Following is the format of the output:

- Metal layer

Node# Layer Net X,Y IRD IRD_diff Resist Resist_diff Width,Length

- Via layer

Node# Layer Net X,Y IRD IRD_diff Resist Resist_diff -,-

- Tap

Node# Tap Net X,Y

- Switch cell

Node# Switch cell='<cell name>' xy=(X,Y) resist=<resistance>

The length and location unit in the report file is um, the resistance unit is Ohm, and the voltage unit is mV.

EXAMPLES

The following example reports the minimum resistance path of U19/VDD PG pin which is connected to the supply net VDD.

```
prompt> open_rail_result RAIL_RESULT_1
prompt> report_rail_minimum_path -cell U19 -net VDD

Cell  : U19
PG Pin: VDD
Node#  Layer   Net   X,Y(um)          IRD(mV)  Diff(mV)    Res(Ohm)   Diff(Ohm)  Width,Length(um)
=============================================================================================================
  46    Tap  VDD   470.700,   0.660     0.000       -         -         -  --
  16  METAL4  VDD   470.700, 127.730     0.498     0.498      1.100     1.100   7.200,127.070
  14   VIA3  VDD   470.700, 127.730     0.498     0.000      1.116     0.016   --
  14  METAL3  VDD   470.700, 127.730       -         -          -         -   --
  12   VIA2  VDD   470.700, 127.730     0.498     0.000      1.131     0.016   --
  12  METAL2  VDD   470.700, 127.730       -         -          -         -   --
  10    VIA  VDD   470.700, 127.730     0.498     0.000      1.145     0.014   --
   0  METAL  VDD   441.120, 127.730     0.499     0.001      3.489     2.344   1.060,29.580
```

The following example reports the minimum resistance path of the switching cell switchcell_class_1.

```
prompt> open_rail_result RAIL_RESULT_1
prompt> report_rail_minimum_path -cell switchcell_class_1 -net VDDINT

Cell  : switchcell_class_1
PG Pin: vddx
Node#  Layer   Net    X,Y(um)          IRD(mV)  Diff(mV)    Res(Ohm)   Diff(Ohm)  Width,Length(um)
=============================================================================================================
  32    Tap  VDD    635.600, 912.551     0.000       -         -         -  --
  28     M6  VDD    673.577, 925.551     0.216     0.216      0.021     0.021   24.000,50.978
  26     V5  VDD    673.577, 925.551     0.335     0.120      0.058     0.037   --
  22     M5  VDD    673.577, 947.551     0.759     0.424      0.248     0.190   3.600,22.000
  20     V4  VDD    673.577, 947.551     0.781     0.022      0.309     0.062   --
```

```
18    M4  VDD    645.078, 947.551    0.810    0.029     0.338     0.029   3.600,28.500
16    V3  VDD    645.078, 947.551    0.893    0.083     0.400     0.062   --
 8    M3  VDD    645.078,1001.288    1.370    0.477     0.864     0.464   3.600,53.736
 6    V2  VDD    645.078,1001.288    1.397    0.027     0.979     0.114   --
 2    M2  VDD    641.150,1001.288    1.443    0.045     1.147     0.168   1.800,3.928
 1  Switch cell='swcell_1' xy=(641.901,1001.288) resist=15.500
 0    M1  VDDINT 642.825,1000.700    4.321    2.878     16.647    15.500   --
```

SEE ALSO

open_rail_result(2)

report_rail_result

Writes rail result data to a text file.

SYNTAX

status **report_rail_result**
-type *type_name*
-supply_nets *supply_net_objects*
[-threshold *threshold*]
[-limit *count*]
file_name

Data Types

string *file_name*
string *type_name*
float *threshold*
integer *count*
supply_net_collection *supply_net_objects*

ARGUMENTS

file_name

Specifies the name of the file to which the rail data is to be saved.

-type *type_name*

Specifies what type of rail data to be reported. The supported data types are pg_pin_power, voltage_drop_or_rise, effective_voltage_drop, minimum_path_resistance, effective_resistance, instance_minimum_path_resistance, instance_power, missing_vias, and unconnected_instances.

-threshold *threshold*

Specifies a threshold to filter the rail data to be reported. When specified, rail data that is smaller than the specified threshold is filtered out.

This option is supported for the following report types: **pg_pin_power**, **voltage_drop_or_rise**, and **effective_voltage_drop**.

-limit *count*

Specifies the number of elements to be reported.

When you use this option, the specified number of elements are included in the file. The elements are sorted in descending order of the value.

-supply_nets *supply_net_objects*

Specifies the supply net objects to be reported.

This option is supported for the following report types: **pg_pin_power**, **voltage_drop_or_rise**, **effective_voltage_drop**, **instance_minimum_path_resistance**, and **and minimum_path_resistance**.

DESCRIPTION

The command writes a file containing rail data from the current result. In the output file, the rail data is listed in a descending order.

The rail data reported to the file can be limited to a specific type by using the **-type** option. The supported types are

- **pg_pin_power**: The power value on each power or ground pin

  The format of the output file is

  PG_PIN full_path_cell_instance_name pg_pin_name power

- **voltage_drop_or_rise**: Dynamic voltage violations

  The format of the output file is

  full_path_cell_instance_name/pg_pin_name voltage

- **effective_voltage_drop**: The effective voltage violations

  For static analysis, the format of the output file is

  cell_instance_pg_pin_name mapped_pg_pin_name supply_net_name effective_voltage_drop
  The output is sorted by **effective_voltage_drop**.

  For dynamic analysis, the format of the output file is
  cell_instance_pg_pin_name mapped_pg_pin_name supply_net_name
  average_effective_voltage_drop_in_tw max_effective_voltage_drop_in_tw
  min_effective_voltage_drop_in_tw max_effective_voltage_drop

  The output is sorted by **max_effective_voltage_drop**.

- **minimum_path_resistance**: Power and ground pin minimum path resistance values

  The format of the output file is

  full_path_cell_instance_name/pg_pin_name resistance

- **instance_minimum_path_resistance**: Minimum path resistance value on each instance

  The format of the output file is

  Rmax(ohm): maximum value of report
  Rmin(ohm): minimum value of report
  Supply_net Total_R(ohm) R_to_power(ohm) R_to_ground(ohm)
  Location Pin_name Instance_name

  The output is sorted by **Total_R**.

- **instance_power**: Power values on each instance

- **missing_vias**: Overlaps of supply net routing on different layers that do not have vias connected
- **unconnected_instances**: Instances that are not connected to any ideal voltage drop sources

The power unit in the report file is Watt, the current unit is Amp, and the voltage unit is Volt.

EXAMPLES

The following example writes a file containing top five PG pin power values to an output file called power.rpt.

```
prompt> open_rail_result RAIL_RESULT_1
prompt> report_rail_result -type pg_pin_power -limit 5 \
  power.rpt -supply_nets { VSS }
1
prompt> sh cat power.rpt
FI2/vss 4.81004e-06
FI6/vss 4.80959e-06
FI3/vss 4.79702e-06
FI4/vss 4.79684e-06
FI1/vss 4.79594e-06
```

The following example writes the calculated minimum path resistances for the VDD net to an output file called minres.rpt.

```
prompt> open_rail_result RAIL_RESULT_1
prompt> report_rail_result -type minimum_path_resistance \
  minres.rpt -supply_nets { vdd }
1
prompt> sh cat minres.rpt
FEED0_14/vbp 518.668
INV61/vdd 158.268
ARCR0_1/vbp 145.582
FEED0_5/vdd 156.228
INV11/vbp 518.669
...
```

The following example writes the calculated effective voltage drop values for the VDD and VSS nets to an output file called inst_effvd.rpt.

```
prompt> open_rail_result RAIL_RESULT_1
prompt> report_rail_result -type effective_voltage_drop \
  -supply_nets { VDD VSS } inst_effvd.rpt
1
prompt> sh cat inst_effvd.rpt
#<cell_instance_pg_pin_name> <mapped_pg_pin_name> <supply_net_name>
#   <average_effective_voltage_drop_in_tw> <max_effective_voltage_drop_in_tw>
#     <min_effective_voltage_drop_in_tw> <max_effective_voltage_drop>
S32_reg_14_/VDD VSS VDD
  -7.750034332e-03 -1.017999649e-02 -5.850076675e-03 -1.029992104e-02
S32_reg_15_/VDD VSS VDD
  -7.709980011e-03 -1.013994217e-02 -5.810022354e-03 -1.026010513e-02
S46_reg_8_/VDD VSS VDD
  -7.179975510e-03 -9.490013123e-03 -5.330085754e-03 -1.021003723e-02
 ...
```

The following example writes the minimum path resistances on cell instances to an output file called inst_minres.rpt.

```
prompt> open_rail_result RAIL_RESULT_1
prompt> report_rail_result -type instance_minimum_path_resistance \
  -supply_nets { VDD VSS } inst_minres.rpt
1
prompt> sh cat inst_minres.rpt
Rmax(ohm) = 162.738
Rmin(ohm) = 4.418
Supply_net   Total_R(ohm)  R_to_power(ohm)  R_to_ground(ohm)
      Location    Pin_name  Instance_name

     VDD        162.738         152.686          10.052
586.000, 496.235       VDDL  ISO_memX_dataout_44__UPF_ISO
     VSS        156.266         137.963          18.303
544.000, 613.235        VSS  GPRs/U2317
     VDD        156.266         137.963          18.303
544.000, 613.235       TVDD  GPRs/U2317
     VDD        152.690         135.223          17.467
602.400, 467.435       VDDL  ISO_memX_dataout_33__UPF_ISO
 ...
```

SEE ALSO

open_rail_result(2)
list_rail_results(2)
close_rail_result(2)

report_rail_scenario

Reports all associations between rail results and design scenarios.

SYNTAX

status **report_rail_scenario**

DESCRIPTION

The command reports all associations between rail results and design scenarios set through the **set_rail_scenario** command.

EXAMPLES

The following example loads three different rail results, and sets associations to design scenarios.

```
prompt> open_rail_result { static_result vcd_result vector_free_result }
prompt> set_rail_scenario -design_scenario fast -result_name
    static_result -percentage_threshold 0.02 -priority_weight 5.0
prompt> set_rail_scenario -design_scenario typical -result_name
    vcd_result -percentage_threshold 0.1 -priority_weight 1.0
prompt> set_rail_scenario -design_scenario slow -result_name
    vector_free_result -percentage 0.1 -priority_weight 1.0
prompt> report_rail_scenario
-------------------------------------------------------------------------------
Design_scenario          Rail_result              Priority  Percentage_threshold
fast                     static                   5         2.00%
typical                  vcd_result               1         10.0%
slow                     vector_free_result       1         10.0%
-------------------------------------------------------------------------------
```

SEE ALSO

open_rail_result(2)
set_rail_scenario(2)

report_rdl_routes

Generates a report of RDL nets.

SYNTAX

int **report_rdl_routes**
[-file *output_file*]
[-open_nets true | false]
[-nets *collection_of_nets* | -nets_in_file *nets_file*]
[-create_error_data false | flyline_only | verification]

Data Types

output_file string
collection_of_nets collection
nets_file string

ARGUMENTS

-file *output_file*

Specifies the name of the output file to which to write the list of RDL nets.

-open_nets true | **false**

If this option is true, the command reports all open specified RDL nets. If this option is false, the command reports all specified RDL nets. By default, **-open_nets** is false.

Nets that are routed with GLinks are not considered to be open nets. GLinks are the result of RDL global routing and are used during RDL detail routing. See **route_rdl_flip_chip** for more information.

-nets *collection_of_nets*

Specifies the flip-chip nets to report.

-nets_in_file *nets_file*

Specifies the name of the file that contains the list of flip-chip nets to report.

-nets has higher priority than **-nets_in_file**.

-create_error_data false | **flyline_only** | **verification**

Specifies the mode of create_error_data. In flyline_only mode, the command will create an error data to save the flylines. In verification mode, the command will check open and short violations and create an error data. For flyline_only/verification mode, user can use error browser to open the error data. In false mode, the command will not create any error data.

The default value is false.

DESCRIPTION

The command displays information about the RDL nets.

EXAMPLES

The following example reports all open RDL nets in the design and writes open net names to the file named RDLopen.

```
prompt> report_rdl_routes  -open_nets true -file RDLopen
```

SEE ALSO

route_rdl_flip_chip(2)
optimize_rdl_routes(2)
push_rdl_routes(2)

report_ref_libs

Gets the reference library list from a library.

SYNTAX

report_ref_libs
[-library *name*]

ARGUMENTS

-library *name*

Which library to generate the reference library report from. If no library is specified, the *current_lib* is used.

DESCRIPTION

The **report_ref_libs** command generates a report listing each library in the reference library path list. Each line reports the library in-memory name (if known), the library path, and the location of the library on disk (if known.)

If the library is open, that is indicated with a "*". If the library has technology information, that is indicated with a "+". If the library has hierarchical reference libraries, that is indicated with a ">", and those reference libraries are indented under this library. A library's in-memory name and location on disk is only given if it is currently open.

EXAMPLES

The following example sets a reference library list on this library, then generates a reference library report.

```
prompt> set_ref_libs -ref_libs {r4000 RAMS macros/TSMC24}
r4000 RAMS macros/TSMC24
prompt> report_ref_libs
****************************************
Report : Reference Library Report
Library: myproj
Version:
Date   : Fri May  7 16:16:08 2010
****************************************
```

```
   Name   Path         Location
   -----------------------------
*   r4000  r4000        /user/joe/deslib/r4000
   -      RAMS         -
*+  TSMC24 macros/TSMC24 /user/common/libcells/macros/TSMC24
   "*" = Library currently open
   "+" = Library has technology information
   "-" = Library name and location not available
1
prompt>
```

The following example shows a reference library with a hierarchical reference library list.

```
prompt> report_ref_libs
****************************************
Report : Reference Library Report
Library: mylib
Version: L-2016.03
Date   : Mon Aug 22 13:03:57 2016
****************************************

   Name     Path         Location
   --------------------------------
*> CORE     CORE         /user/joe/deslib/CORE
*    ALU    ALU          /user/joe/deslib/ALU
*    CNTRL  CNTRL        /user/joe/deslib/CNTRL
*+   TSMC24 macros/TSMC24 /user/common/libcells/macros/TSMC24
*   RAMS    RAMS         /user/common/RAMS
*+  TSMC24   macros/TSMC24 /user/common/libcells/macros/TSMC24
   "*" = Library currently open
   "+" = Library has technology information
   ">" = Library has hierarchical reference libraries
1
prompt>
```

SEE ALSO

create_lib(2)
current_lib(2)
get_attribute(2)
set_ref_libs(2)
lib_attributes(3)
search_path(3)

report_reference

Reports the references in the current instance or design.

SYNTAX

status **report_references**
[-nosplit]
[-hierarchical]
[-by_module]

ARGUMENTS

-nosplit

Does not split lines if the column overflows.

-hierarchical

Reports all the references inside the logic hierarchy.

-by_module

This option can be used with **hierarchical** option only, it reports the references based on module and from the full design hierarchy.

DESCRIPTION

Displays information about all references in the current instance or current design. If the current instance is set, the report is generated relative to that instance; otherwise, the report is generated for the current design.

EXAMPLES

The following is an example of the report generated by the **report_references** command.

prompt> **report_references**

****
Report : reference

```
Design : middle
****************************************

Attributes:
   b - black-box (unknown)
   h - hierarchical
   n - noncombinational
   u - contains unmapped logic
   E - extracted timing model
   Q - Quick timing model (QTM)

Reference      Library      Unit Area  Count  Total Area   Attributes
--------------------------------------------------------------------------
FD2            tech_lib         3.00      4       12.00   b
ND2            tech_lib         3.00      4       12.00   b
inter                           6.00      2       12.00   h
low                             2.00      2        4.00   h
--------------------------------------------------------------------------
Total 4 references                                 40.00
```

The following example uses the **-by_module** option with **-hierarchical** option in report_references to generate report:

```
prompt> report_references -by_module -hierarchical
****************************************
Report : reference
       -hierarchical_info
Design : r4000
Version: N-2017.09-SP4-BETA
Date   : Sun Nov 19 22:20:32 2017
****************************************

Attributes
   b - black-box (unknown)
   h - hierarchical
   n - noncombinational
   u - contains unmapped logic
   E - extracted timing model
   Q - Quick timing model (QTM)


Reference        Library      Unit Area   Count    Total Area   Attributes
--------------------------------------------------------------------------------
AN3              tech_lib      2.00          8     16.00
mid_0                         56.00          1     56.00      h, n
--------------------------------------------------------------------------------
Total 2 references                                   72.00

****************************************
Design: mid_0
****************************************
Reference        Library      Unit Area   Count    Total Area   Attributes
--------------------------------------------------------------------------------
low_0                         28.00          1     28.00      h, n
low_5                         28.00          1     28.00      h, n
--------------------------------------------------------------------------------
Total 2 references                                   56.00
```

```
****************************************
Design: low_0
****************************************
Reference       Library     Unit Area   Count    Total Area   Attributes
--------------------------------------------------------------------------------
FD1             tech_lib     7.00        4     28.00     n
--------------------------------------------------------------------------------
Total 1 references                          28.00

****************************************
Design: low_5
****************************************
Reference       Library     Unit Area   Count    Total Area   Attributes
--------------------------------------------------------------------------------
FD1             tech_lib     7.00        4     28.00     n
--------------------------------------------------------------------------------
Total 1 references                          28.00
```

SEE ALSO

report_hierarchy(2)
report_cells(2)

report_references

Reports the references in the current instance or design.

SYNTAX

status **report_references**
[-nosplit]
[-hierarchical]
[-by_module]

ARGUMENTS

-nosplit

Does not split lines if the column overflows.

-hierarchical

Reports all the references inside the logic hierarchy.

-by_module

This option can be used with **hierarchical** option only, it reports the references based on module and from the full design hierarchy.

DESCRIPTION

Displays information about all references in the current instance or current design. If the current instance is set, the report is generated relative to that instance; otherwise, the report is generated for the current design.

EXAMPLES

The following is an example of the report generated by the **report_references** command.

prompt> **report_references**

****
Report : reference

```
Design : middle
****************************************

Attributes:
   b - black-box (unknown)
   h - hierarchical
   n - noncombinational
   u - contains unmapped logic
   E - extracted timing model
   Q - Quick timing model (QTM)

Reference      Library      Unit Area  Count  Total Area   Attributes
--------------------------------------------------------------------------
FD2            tech_lib         3.00      4       12.00   b
ND2            tech_lib         3.00      4       12.00   b
inter                           6.00      2       12.00   h
low                             2.00      2        4.00   h
--------------------------------------------------------------------------
Total 4 references                                 40.00
```

The following example uses the **-by_module** option with **-hierarchical** option in report_references to generate report:

```
prompt> report_references -by_module -hierarchical
****************************************
Report : reference
      -hierarchical_info
Design : r4000
Version: N-2017.09-SP4-BETA
Date   : Sun Nov 19 22:20:32 2017
****************************************

Attributes
   b - black-box (unknown)
   h - hierarchical
   n - noncombinational
   u - contains unmapped logic
   E - extracted timing model
   Q - Quick timing model (QTM)


Reference       Library      Unit Area   Count    Total Area   Attributes
--------------------------------------------------------------------------
AN3             tech_lib      2.00          8     16.00
mid_0                        56.00          1     56.00      h, n
--------------------------------------------------------------------------
Total 2 references                                 72.00

****************************************
Design: mid_0
****************************************
Reference       Library      Unit Area   Count    Total Area   Attributes
--------------------------------------------------------------------------
low_0                        28.00          1     28.00      h, n
low_5                        28.00          1     28.00      h, n
--------------------------------------------------------------------------
Total 2 references                                 56.00
```

```
****************************************
Design: low_0
****************************************
Reference       Library     Unit Area   Count    Total Area   Attributes
--------------------------------------------------------------------------------
FD1             tech_lib     7.00        4     28.00     n
--------------------------------------------------------------------------------
Total 1 references                          28.00

****************************************
Design: low_5
****************************************
Reference       Library     Unit Area   Count    Total Area   Attributes
--------------------------------------------------------------------------------
FD1             tech_lib     7.00        4     28.00     n
--------------------------------------------------------------------------------
Total 1 references                          28.00
```

SEE ALSO

report_hierarchy(2)
report_cells(2)

report_regular_multisource_clock_tree_options

It reports options from command set_regular_multisource_clock_tree_options for performing auto tap synthesis and global clock tree synthesis for regular multisource clock trees.

SYNTAX

status **report_regular_multisource_clock_tree_options**

DESCRIPTION

The *report_regular_multisource_clock_tree_options* command reports options set through the corresponding set option command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example sets the auto tap synthesis options and reports the options.

```
prompt> set_regular_multisource_clock_tree_options -clock clk -topology htree_only -prefix MSCTS \
 -tap_boxes {4 2} -tap_lib_cells [get_lib_cells */CKBUF*] \
 -htree_lib_cells [get_lib_cells */CKINV*] -htree_layers "m9 m10" -htree_routing_rule "htree_ndr"

prompt> report_regular_multisource_clock_tree_options
```

SEE ALSO

synthesize_regular_multisouce_clock_trees(2)
set_regular_multisouce_clock_tree_options(2)
remove_regular_multisouce_clock_tree_options(2)

report_repeater_group_constraints

Reports constraints that are set for a group of repeaters.

SYNTAX

status **report_repeater_group_constraints**

DESCRIPTION

This command reports the constraints for a group of repeaters that were set using place_group_repeaters, add_group_repeaters, or set_repeater_group_constraints commands. The constraints include horizontal_repeaters_spacing, vertical_repeater_spacing and layer_cutting_distance.

EXAMPLES

The following examples show the usages of this command.

```
prompt> set_repeater_group_constraints -type insertion -horizontal_repeater_spacing 7
-layer_cutting_distance {{M1 2.12 13.2} {M2 0.5 0.78} {M3 3.01 3.78}}
prompt> set_repeater_group_constraints -type placement -horizontal_repeater_spacing 10
prompt> report_repeater_group_constraints

Report Format:

Type:  Insertion
-------------------------------------------------
 Horizontal repeater spacing: 7
 Vertical repeater spacing: -
 Layer cutting distance: { { M1 2.120000 13.200000 }  { M2 0.500000 0.780000 }
 { M3 3.010000 3.780000 }  }

 -------------------------------------------------

 Type:  Placement
-------------------------------------------------
 Horizontal repeater spacing: 10
 Vertical repeater spacing: -
 Layer cutting distance: { }

 -------------------------------------------------
```

SEE ALSO

set_repeater_group_constraints(2)
remove_repeater_group_constraints(2)

report_repeater_groups

Report infomation of repeater groups.

SYNTAX

status **report_repeater_groups**
[-group_ids *group_id_list*]

Data Types

group_id_list list

ARGUMENTS

-group_ids *group_id_list*

Specifies list of group ids to report. Report all groups if the option is not specified.

DESCRIPTION

This command reports cells and cutline of each register group, and path drivers and path loads if there is any.

EXAMPLES

The following examples show the usages of command.

prompt> **report_repeater_groups**

prompt> **report_repeater_groups -group_ids {1 2}**

SEE ALSO

set_repeater_group(2)
place_group_repeaters(2)

report_resources

Lists the resources and datapath blocks used in the design.

SYNTAX

string **report_resources**
[-summary]
[-power]
[-toggling_activity]
[-html_file *file*]
[-hierarchy]
[-nosplit]
[*module_list*]

ARGUMENTS

-summary

Reports the summary of singleton and extracted datapath of the design or specified modules. The implementation overview is also reported. Without specifying this option, all resources are reported in detail view that each cell is in separate section.

-power

Reports the information related to power, including internal, switching, leakage and total power for each DesignWare cell that is not ungrouped. When this options is set, the command may take longer because of the propagation of switching activity.

-toggling_activity

Reports the toggling activity of each inputs of the datapath, in detail view. This option can't be used together with **-summary**.

-html_file *file*

Redirect the output of the report to specified files in HTML format.

-hierarchy

Reports information about all resources used in the hierarchy of specified modules. This option can only be used when *module_list* is specified. By default, the tool reports all resources hierarchically if *module_list* is not specified; or only the resources in current hierarchy if *moulde_list* is specified.

-nosplit

Does not split lines if column overflows.

module_list

The list of modules for which the resources are reported. When not specified, all hierarchies of current design is reported.

DESCRIPTION

This command lists the resources and datapath blocks used in the NDM design after compile.

A resource is an arithmetic or comparison operator read in as part of an HDL design. Resources can be shared when compiling the design and are reported after the compile operation completes.

A datapath block contains one or more resources that are grouped together and optimized by a datapath generator.

This report also has information on the parameters used to build the module, user-declared resources in each module, shared operations, and resources that are transformed into datapath blocks.

If the resources are not ungrouped after compile, this report will additionally report the area, power, slack information. Specially, the power is reported only when the leakage and dynamic scenarios can be automatically recognized.

EXAMPLES

The following example displays resource information for the current design. The content summary section has three tables: singleton resources width, extracted resources width, and a statisitics table. The implementation overview section lists all resources with their module name, current implementation and multiplier arch info.

```
prompt> report_resources -summary
****************************************
Report : report_resources
Design : test
Version: 2.5.5
Date   : Tue Mar 28 01:25:45 2017
****************************************
-----------------
 Content Summary
-----------------

Number of synthetic cells:                26
Number of singleton:                      17
Number of datapath:                       9
Number of "area" opt:                     24
Number of "speed" opt:                    2


 Singleton            Count  Min Width  Max Width      Ave.
--------------------------------------------------------------------
 DW01_add               8        1        16        5.5
 DW01_ash               3        9        12       11.0
 DW01_cmp2              1       21        21       21.0
 DW01_sub               2        8         8        8.0
 DW02_multp             1       16        16       16.0
 DW_div                 1        7         7        7.0
 DW_div_uns             1        7         7        7.0
```

```
 Extracted            Count  Min Width  Max Width      Ave.
-------------------------------------------------------------------------
 *                    3        6        11       9.3
 + -                  25       6        12       8.7
 == !=                 2       4         6       5.0
 ?                    1        7        7        7.0

-------------------------
 Implementation Overview
-------------------------

                                                  Multiplier
 Cell                 Module                      Implementation  Arch
-------------------------------------------------------------------------------
 DP_OP_37_35775_64218_J2   DP_OP_37_35775_64218_J2  str(speed)
 DP_OP_38_16020_64218_J2   DP_OP_38_16020_64218_J2  str(area)
 DP_OP_40_27026_64218_J2   DP_OP_40_27026_64218_J2  str(area)      benc_radix4
 RS_OP_41_7276_64218_J2/
  DP_OP_36_25796_27024_J0 DP_OP_36_25796_27024_J0  str(area)
 U0/DP_OP_16_1311_65259_J1  DP_OP_16_1311_65259_J1   str(area)
 U0/DP_OP_17_20566_65259_J1 DP_OP_17_20566_65259_J1  str(area)      benc_radix4
 U0/DP_OP_18_13129_65259_J1 DP_OP_18_13129_65259_J1  str(area)
 U0/sll_28             DW01_ash              str(area)
 U0/sub_25             DW01_sub              pparch(area)
 U2/DP_OP_11_46531_62518_J3 DP_OP_11_46531_62518_J3  str(area)
 U2/U1/
  DP_OP_16_1311_65259_J1   DP_OP_16_1311_65259_J1   str(area)
 U2/U1/
  DP_OP_17_20566_65259_J1  DP_OP_17_20566_65259_J1  str(area)      benc_radix4
 U2/U1/
  DP_OP_18_13129_65259_J1  DP_OP_18_13129_65259_J1  str(area)
 U2/U1/sll_28           DW01_ash              str(area)
 U2/U1/sub_25           DW01_sub              pparch(area)
 U2/add_46              DW01_add              pparch(area)
 U2/sll_41              DW01_ash              str(area)
 U3/DP_OP_8_19016_6742_J4  DP_OP_8_19016_6742_J4   str(area)
 add_96                 DW01_add              pparch(speed)
 div_126                DW_div_uns            cla(area)
 div_126/u_div          DW_div                cla(area)
 div_126/u_div/
  u_add_PartRem_1_0      DW01_add              pparch(area)
 div_126/u_div/
  u_add_PartRem_1_1      DW01_add              pparch(area)
 div_126/u_div/
  u_add_PartRem_1_2      DW01_add              pparch(area)
 div_126/u_div/
  u_add_PartRem_1_3      DW01_add              pparch(area)
 div_126/u_div/
  u_add_PartRem_1_4      DW01_add              pparch(area)
 div_126/u_div/
  u_add_PartRem_1_5      DW01_add              pparch(area)
 lte_99                 DW01_cmp2             pparch(area)
 multp_U                DW02_multp            pparch(area)    and
```

The following example shows the resource report for specified modules *dp_block_s_0*. As shown in the example, for each datapath cell in the specified module, the module name, parameters, implementation, toggling activity and detailed datapath expressions etc.

are presented in a easy to read manner. Note that the impelmentation set by **set_implementation** command and the architecture options set by **set_datapath_architecture_options** command will also be reported if available.

```
prompt> report_resources -toggling_activity -nosplit dp_blocks_s_0
****************************************
Report : report_resources
Design : test
Version: 2.5.5
Date   : Fri Mar 31 20:47:51 2017
****************************************
----------------------
 Design : dp_block_s_0
----------------------

 Cell : U2/U1/sub_25
---------------------

  Module             : DW01_sub
  Parameters         : width=8
  Current Implementation:  pparch(area)
  Contained Operations  :  sub_25(dev.v:25)

  Input                              With
  Ports  Width  Variable Inputs  Switching Activity
 ------------------------------------------------------
  A        8       0    0%          0    0%
  B        8       7   88%          0    0%
  CI       1       0    0%          0    0%

 Cell : U2/U1/DP_OP_16_1311_65259_J1
-------------------------------------

  Current Implementation:  str(area)
  Contained Operations  :  add_28(dev.v:28) add_28_2(dev.v:28) add_28_3(dev.v:28)

            Data
  Var     Type  Class     Width  Expression
 ------------------------------------------------------------------------------
  PI_0    PI   Unsigned     11
  PI_1    PI   Unsigned      7
  PI_2    PI   Unsigned      9
  PI_3    PI   Unsigned     12
  PO_0    PO   Unsigned     12  PI_0 + PI_1 + PI_2 + PI_3 (dev.v:28)

  Input                              With
  Ports  Width  Variable Inputs  Switching Activity
 ------------------------------------------------------
  PI_0     11       11   100%          0    0%
  PI_1      7       7   100%          0    0%
  PI_2      9       9   100%          0    0%
  PI_3     12       12   100%          0    0%
```

SEE ALSO

compile(2)
report_references(2)
set_implementation(2)
set_datapath_architecture_options(2)

report_routing_corridors

Reports the corridor names and associated shapes, min and max layers, nets and supernets for the specified routing corridors.

SYNTAX

status **report_routing_corridors**
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[-output *file_name*]
[*routing_corridor_list*]

Data Types

digits int
file_name string
routing_corridor_list list

ARGUMENTS

-verbose

Displays additional debugging information.

-nosplit

Writes out long report lines, even if the line length is greater than 80 columns. Use this option to simplify post-processing of the output or when comparing the report files by using the UNIX **diff** command. If this option is not specified, the command breaks long constraint command lines near column 80 and inserts a line continuation character (\).

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13; the default is determined by the **shell.report_default_significant_digits** application option, whose default is 2. Use this option to override the default.

routing_corridor_list

Specifies the routing corridors on which to report. The list can contain routing corridor names, patterns, or collections. A collection can be specified by using the **get_routing_corridors** command. If you do not specify this option, all routing corridors in the design are reported.

-output *file_name*

Generates a Tcl script that contains **create_routing_corridor** commands. You can use the Tcl script to re-create the existing routing corridors.

DESCRIPTION

This command reports information about the specified routing corridors, including

- The name of the routing corridor
- The names, minimum and maximum layers, of the corridor shapes contained in the routing corridor
- The names of the associated nets and supernets

This command also reports the results of connectivity checking and pin checking.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports information about the routing corridor named RC_1.

```
prompt> report_routing_corridors RC_1

CORRIDOR NAME: RC_1
Shape Name              min/max     Shape
--------------------------------------------------------------------------------
CORRIDOR_SHAPE_1        M2/M5       {20.00 20.00} {30.00 21.00}
CORRIDOR_SHAPE_2        M4/M6       {25.00 25.00} {27.00 30.00}
--------------------------------------------------------------------------------
Shapes Connected:         no
Shapes Cover Pins and Ports: no
Objects:                n4 supernet_1
--------------------------------------------------------------------------------
```

SEE ALSO

add_to_routing_corridor(2)
check_routing_corridors(2)
create_routing_corridor(2)
create_routing_corridor_shape(2)
get_routing_corridor_shapes(2)
get_routing_corridors(2)
remove_from_routing_corridor(2)
remove_routing_corridor_shapes(2)
remove_routing_corridors(2)
report_routing_corridors(2)

report_routing_guides

Reports details of access preference type of routing guides.

SYNTAX

status **report_routing_guides**
-rectangle *rect*
[-level *all* | *top_level*]

Data Types

rect bounding box points

ARGUMENTS

-rectangle *rect*

Specifies the coordinates of the bounding box of the routing guide for which we want the details (including the access preference data).

This option is required.

-level *all* | *top_level*

Specifies the level on which the routing guides are to be searched.

If you do not specify this option, the tool searches at all levels hierarchically. You can specify "top_level" value for this option if you want the top level route guides to be searched only.

DESCRIPTION

This command fetches the routing guides based on the bbox specified by the user as part of the -rectangle option. The routing guides whose base rectangle overlaps with the bbox specified are considered for printing out details.

The details include the access preference related data.

The returned values are plain text information.

EXAMPLES

The following examples depict the behavior.

```
prompt> report_routing_guides -rectangle {{0.0 0.0} {100.0 100.0}} \
      -level top_level


Info for Unnamed Route Guide
Access Preference Data for Layer - METAL4
  wire_access_preference  Rect - (100000, 100000 -> 200000, 200000)      Strength - 1.000000
  wire_access_preference  Rect - (1400000, 1400000 -> 1500000, 1500000)   Strength - 1.000000
  via_access_preference   Rect - (100000, 100000 -> 200000, 200000)      Strength - 1.000000
  via_access_preference   Rect - (1200000, 1200000 -> 1500000, 1400000)   Strength - 1.000000
  .in -0.25in
```

SEE ALSO

get_routing_guides(2)
create_routing_guide(2)

report_routing_rules

Reports non-default routing rules in the design.

SYNTAX

status **report_routing_rules**
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[-output *file_name*]
[-of_objects *net_list*]
[*routing_rule_list*]

Data Types

digits int
file_name string
net_list list
routing_rule_list string

ARGUMENTS

-verbose

Displays verbose non-default rule information.

-nosplit

Does not split lines if column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default. Fixed-precision floating-point arithmetics is used for delay calculation; therefore, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h. The upper bound on a meaningful value of the argument to -significant_digits is then FLT_DIG -ceil(log10(fabs(any_reported_value))).

-output *file_name*

Generates a Tcl script that defines the specified routing rules. If the *routing_rule_list* option is used, the Tcl script will contain the non-default rule definitions. If **-of_objects** *net_list* option is used, the Tcl script will contain the net non-default rule assignments.

-of_objects *net_list*

Specifies the nets of which to report non-default rules.

routing_rule_list

Specifies the non-default routing rules to report. By default, the tool reports all non-default routing rules in the design.

DESCRIPTION

This command reports the details of the specified non-default routing rules, or the non-default routing rules assigned to the specified nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports on the non-default rule named WideMetal.

```
prompt> report_routing_rules WideMetal
```

SEE ALSO

create_routing_rule(2)
remove_routing_rules(2)
set_routing_rule(2)

report_rp_groups

Reports the placement status of relative placement groups and violations of relative placement constraints.

SYNTAX

collection **report_rp_groups**
{*rp_group_list* | -all}
[-critical]
[-unplaced]
[-non_critical]
[-verbose]

Data Types

rp_group_list list or collection

ARGUMENTS

rp_group_list

Specifies the relative placement groups that will be checked and reported. This option is mutually exclusive with **-all**.

-all

Specifies that all relative placement groups should be checked and reported. This option is mutually exclusive with *rp_group_list*.

-critical

Specifies to report the relative placement groups that could not be placed.

-unplaced

Specifies to report the relative placement groups that are unplaced. The placement of such relative placement groups has not been done.

-non_critical

Specifies to report the relative placement groups that are placed but not meeting relative placement constraints.

-verbose

Specifies to report exact location of the failures in relative placement group.

DESCRIPTION

The **report_rp_groups** command checks and reports any violation of constraints for the specified relative placement groups along with its placement status. The relative placement group placement status and violations are reported in following four categories for top level relative placement groups.

The placed relative placement groups without any constraint violations are reported under category *placed top level relative placement groups*.

The placed relative placement groups with constraint violations are reported under category *top level groups not meeting constraints, but placed*.

The relative placement groups which are not placed yet are reported under category *unplaced top level relative placement groups*.

The relative placement groups which could not be placed successfully are reported under category *failed top level relative placement groups*.

The command returns a collection containing the relative placement groups that are checked. If no groups are checked, an empty string is returned.

EXAMPLES

The following examples use **report_rp_groups** to report constraint violations:

```
prompt> report_rp_groups rp_fixed_cell

****************************************************************************
Report  : Relative Placement Summary
Total number of specified top level relative placement groups: 1
Total number of placed top level relative placement groups: 1
Total number of top level groups not meeting constraints, but placed: 1
****************************************************************************

****************************************************************************
Report: top level relative placement groups, not meeting all constraints, but
placed
****************************************************************************
RP Group: rp_fixed_cell
----------------------------------------------------------------------------
Warning: The 'column' alignment has not been respected for relative placement
group. (RPGP-004)
{rp_fixed_cell}

prompt> report_rp_groups rp_fixed_cell -non_critical -verbose

****************************************************************************
Report  : Relative Placement Summary
Total number of specified top level relative placement groups: 1
Total number of placed top level relative placement groups: 1
Total number of top level groups not meeting constraints, but placed: 1
****************************************************************************

****************************************************************************
```

```
Report: top level relative placement groups, not meeting all constraints, but
placed
****************************************************************************
RP Group: rp_fixed_cell
----------------------------------------------------------------------------
Warning: The 'column' alignment has not been respected for relative placement
group. (RPGP-004)
Location details of alignment failure:
----------------------------------------------------------------------------
RP Group                        Row     Column    Failed Alignment
----------------------------------------------------------------------------
rp_fixed_cell                   0       1         left
----------------------------------------------------------------------------
{rp_fixed_cell}

prompt> report_rp_groups -all -unplaced

****************************************************************************
Report  : Relative Placement Summary
Total number of specified top level relative placement groups: 4
Total number of placed top level relative placement groups: 2
Total number of unplaced top level relative placement groups: 1
Total number of critical (failed) top level relative placement groups: 1
****************************************************************************

****************************************************************************
Report: unplaced top level relative placement groups
****************************************************************************
RP Group: rp_anchor1
----------------------------------------------------------------------------
Warning: The relative placement group has not been placed yet. (RPGP-003)
{rp_anchor1}

prompt> report_rp_groups -all -critical

****************************************************************************
Report  : Relative Placement Summary
Total number of specified top level relative placement groups: 4
Total number of placed top level relative placement groups: 2
Total number of unplaced top level relative placement groups: 1
Total number of critical (failed) top level relative placement groups: 1
****************************************************************************

****************************************************************************
Report: failed top level relative placement groups (critical)
****************************************************************************
RP Group: rp_me
----------------------------------------------------------------------------
Error: Possible placement failure of relative placement group. (RPGP-002)
{rp_me}
```

SEE ALSO

add_to_rp_group(2)
create_rp_group(2)
remove_rp_groups(2)
check_rp_constraints(2)

report_sadp_track_rule

Reports a track rule.

SYNTAX

status **report_sadp_track_rule**
rules_names | -all
[-nosplit]
[-significant_digits *digits*]

Data Types

rules_names string
digits integer

ARGUMENTS

rule_names

Specifies the list of SADP track rules to report. You must specify one or more SADP track rule names or the **-all** option.

-all

Reports all SADP track rules. You must specify one or more SADP track rule names or the **-all** option.

-nosplit

Specify nosplit style reporting behavior. By default, line wrapping of long rows of data is done automatically. In nosplit mode, line wrapping is not done. Readable column alignment is still maintained in nosplit mode.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 through 13. The default is 2.

DESCRIPTION

This command reports self-aligned double patterning (SADP) track rules set with the **create_sadp_track_rule** command.

EXAMPLES

The following example defines three SADP track rules in the current library and reports them.

```
prompt> create_sadp_track_rule -name tr1 \
  -pattern {{1 1 "vdd"} {2 1 "vss"} } -sadp_spacing 0.50
prompt> create_sadp_track_rule -name tr2 \
  -pattern {{2 1 "vdd"} {1 1 "vss"} } -sadp_spacing 0.50
prompt> create_sadp_track_rule -name tr3 \
  -pattern {{2 1 "vdd"} {2 1 "vss"} } -sadp_spacing 0.50

prompt> report_sadp_track_rule -all
****************************************
Report : report_sadp_track_rule
Lib    : r4000
Version:
Date   :
****************************************
Rule_name  Spacing  Pattern
-----------------------------------------------------
tr1        0.50     { {1.00 1 "vdd"} {2.00 1 "vss"} }
tr2        0.50     { {2.00 1 "vdd"} {1.00 1 "vss"} }
tr3        0.50     { {2.00 1 "vdd"} {2.00 1 "vss"} }
1
```

The following example reports the SADP track rules only for rules tr2 and tr3.

```
prompt> report_sadp_track_rule {tr2 tr3}
****************************************
Report : report_sadp_track_rule
Lib    : r4000
Version:
Date   :
****************************************
Rule_name  Spacing  Pattern
-----------------------------------------------------
tr2        0.50     { {2.00 1 "vdd"} {1.00 1 "vss"} }
tr3        0.50     { {2.00 1 "vdd"} {2.00 1 "vss"} }
```

SEE ALSO

check_sadp_tracks(2)
create_sadp_track_rule(2)
generate_sadp_tracks(2)
remove_sadp_track_rule(2)

report_safety_logic_port_map

Reports the port mapping information for the module or lib-cell used for voting logic in functional safety flow.

SYNTAX

status **report_safety_logic_port_map**
[-module *module_list*]
[-mapping *lib_cell_list*]

Data Types

module_list collection
lib_cell_list collection

ARGUMENTS

-module *module_list*

Specifies one or more the modules for which the port mapping is to be reported. Either objects or names can be specified.

-mapping *lib_cell_list*

Specifies one or more libcells for which the port mapping is to be reported. Either objects or names can be specified.

DESCRIPTION

This command reports the port mapping information of the given module or libcells. If the port map has been set earlier using set_safety_logic_port_map command, then the list of input ports and error and/or voting outputs are printed. If the port map has not been set, then the reporting is skipped. Either -module or -mapping must be specified. Both options can be specified together as well.

EXAMPLES

The following example reports safety logic port map for a lib-cell.

prompt> **report_safety_logic_port_map -mapping MAJ3**

The following example reports safety logic port map for a module.

```
prompt> report_safety_logic_port_map -module DMR_LOGIC
```

SEE ALSO

report_safety_logic_port_map(2)
create_safety_register_rule(2)
write_safety_register_script(2)

report_safety_register_groups

Reports safety register groups of the current design.

SYNTAX

status **report_safety_register_groups**
[*objects*]

Data Types

objects collection

ARGUMENTS

objects

Specifies the group objects or names which need to be reported

DESCRIPTION

Prints a report of the specified safety register group objects. If nothing is specified, it reports all the groups in the current design.

EXAMPLES

The following example creates a collection of safety_register_group objects and then reports them.

prompt> **report_safety_register_groups [get_safety_register_groups group1*]**

SEE ALSO

create_safety_register_group(2)
get_safety_register_groups(2)
write_safety_register_script(2)

report_safety_register_rules

Reports safety register rules from the current design.

SYNTAX

status **report_safety_register_rules**
[*objects*]

Data Types

objects	collection

ARGUMENTS

objects

Specifies the rule objects or names which need to be reported

DESCRIPTION

Prints a report of the specified safety register rule objects. If nothing is specified, it reports all the rules in the current design.

EXAMPLES

The following example creates a collection of safety_register_rule objects and then reports them.

prompt> **report_safety_register_rules [get_safety_register_rules rule1*]**

SEE ALSO

create_safety_register_rule(2)
get_safety_register_rules(2)
write_safety_register_script(2)

report_safety_status

Perform and report checks related to safety_register_rules and safety_register_groups.

SYNTAX

string **report_safety_status**
[-header *string*]
[-safety_register_rules *safety_register_rules*]
[-safety_register_groups *safety_register_groups*]
[-repelling_group_bounds *repelling_group_bounds*]
[-summary]
[-verbose]

string *string*
list *safety_register_rules*
list *safety_register_groups*
list *repelling_group_bounds*

ARGUMENTS

-header *string*

Provide a custom header for the printed report.

-safety_register_rules *safety_register_rules*

Report given safety_register_rule objects. This option restricts the checks and reporting to the provided rules.

-safety_register_groups *safety_register_groups*

Report given safety_register_group objects. This option restricts the checks and reporting to the provided groups.

-repelling_group_bounds *repelling_group_bounds*

Report given repelling_group_bound objects. This option runs the checks and reporting for the provided bounds.

-summary

With this option specified, the report only displays its summary section with statistics of the current design state. The actual messages informing about individual issues will not be printed. The option -summary is mutually exclusive to the -verbose option.

-verbose

Print additional messages such as SR-036 showing the distance between safety registers that satisfy their spacing requirements. By default this information is omitted, and only violations are displayed via SR-007. The option -verbose is mutually exclusive to the -summary option.

DESCRIPTION

The command **report_safety_status** can be run throughout the Synopsys Automotive Flow and performs checks related to safety rules and groups defined in the current block.

The results of the checks are reported in textual format and stored into a message database, called safety_status.ems. This message database provides interactive detail for examining and addressing safety related issues found by **report_safety_status**.

gui_start
open_ems_database safety_status.ems

Then, use **Window->Message Browser Window to examine the interactive report. Messages generated by this report are tagged by SR-???.**

Checks are enabled by default, but it is possible to disable a specific check, for example **SR-005**, as follows:

set_attribute [get_ems_rules {SR-005}] -name is_enabled -value false

Checks performed by this command are:

SR-000 (info) Safety register related object counts for block %block.
SR-001 (warning) Found safety_register_rule %rule without associated safety_register_group.
SR-002 (error) The design still contains a safety critical register (%safety_critical_register) that must be replaced by fault tolerant or re
SR-003 (error) The safety register groups {%list_of_groups} are not mutually exclusive. The shared cells are {%list_of_cells}.
SR-004 (error) The safety_register_group %group has %actual_count redundancy register(s), while %required_count are required b
SR-005 (error) Redundancy register %redundancy_register in safety_register_group %group does not have tap cells placed adjace
SR-006 (error) Safety_register_group %group contains tap cells {%list_of_tap_cells} not referring any of the allowed tap_mapping o
SR-007 (error) Insufficient distance (%actual_distance) between redundancy registers {%list_of_regs} within safety_register_group %
SR-008 (warning) The safety_register_rule %rule has no logic module defined; skipping logic cell checks for dependent groups.
SR-009 (error) Found logic {%list_of_cells} within safety_register_group %group that have no equivalent cell in logic module %logic
SR-010 (error) Found a mismatch between the split pin types in safety_register_rule %rule and associated split pins {%list_pins} in s
SR-011 (error) Split pin %pin in safety_register_group %group still has to be buffered.
SR-012 (error) Missing logic {%list_of_cells} in safety_register_group %group.
SR-013 (error) Split pin %pin without non-buffer driver in safety_register_group %group.
SR-014 (error) Cannot match split pin %pin on redundancy register %redundancy_register in safety_register_group %group.
SR-015 (error) Detected loop in fan-in of split pin %pin within safety_register_group %group.
SR-016 (error) Found buffer driving multiple split pins {%list_pins} within safety_register_group %group.
SR-017 (error) Did not find a common non-buffer driver for matching split pins {%list_pins} within safety_register_group %group.
SR-018 (error) Found register %reg to be associated with safety_register_rule %rule as well as safety_register_group %group.
SR-019 (error) Safety_register_group (%group) for fault_tolerant rule %rule has logic (%list_of_cells).
SR-020 (error) Found cells {%list_of_cells} within safety_register_group %group that have no size_only (reason: safety) to protect th
SR-021 (error) Safety_register_group %group contains registers {%list_of_regs} not referring to the lib_cell specified by the register_
SR-023 (error) The repelling group bound %rgb contains %count cells that violated the requirement of minimum distance between d
SR-024 (error) The repelling group bound %rgb has an invalid number of cores. %core_count
SR-025 (error) The repelling group bound %rgb contains core %ancestor which is an ancestor of %core which is not allowed.
SR-028 (error) Could not find routing blockages with blockage_group_id %rbid for core %core in repelling group bound %rgb.
SR-029 (error) Missing routing blockages on layer %layer with blockage_group_id %rbid for core %core in repelling group bound %
SR-032 (info) Safety repelling group bound related object counts for block %block.
SR-033 (info) Legalization bound for core %core in repelling group bound %rgb.
SR-034 (error) Could not extract a legalization bound for core %core in repelling group bound %rgb.
SR-035 (error) Found group %group with redundancy registers (%list_of_regs) in distinct voltage areas (%list_of_vas).
SR-036 (info) The distance (%actual_distance) between redundancy registers {%list_of_regs} within safety_register_group %group
SR-037 (error) Found group %group with redundancy registers (%list_of_regs) with distinct retention strategies (%list_of_rets).
SR-038 (error) Found unconnected output pins {%list_of_pins} within safety_register_group %group.

SR-039 (error) A safety_register_rule of type fault_tolerant (%rule) should not have any split pin types.
SR-040 (error) A safety_register_group (%group) for a safety_register_rule of type fault_tolerant (%rule) should not have any split pi
SR-041 (error) Found group %group with redundancy registers (%list_of_regs) in distinct power domains (%list_of_pds).
SR-042 (error) Core %core in repelling group bound %rgb has %count internal nets that violate the required net separation (guardb
SR-043 (info) Core %core in repelling group bound %rgb has %count protruding internal nets overlapping routing blockage (blocka
SR-044 (info) Core %core in repelling group bound %rgb has %count protruding transit nets that overlap routing blockage (blockag
SR-045 (warning) Found %cell_count internal cells of core %core blocked by routing blockage (blockage_group_id %rbid) on layer %
SR-046 (warning) Found %cell_count exposed hostile cells from core %other_core, not fully covered by routing blockage (blockage_
SR-047 (error) Found an internal net of core %core in repelling group bound %rgb has a blockage_group_id %rbid which is not asso
SR-048 (error) Found an internal net of core %core in repelling group bound %rgb which has a blockage_group_id %rbid that is ass
SR-049 (error) Found an internal net (%net) of core %core in repelling group bound %rgb is missing a blockage_group_id associate
SR-050 (error) The internal nets of core %core in repelling group bound %rgb are referring to multiple blockage_group_ids {%list_of
SR-051 (info) Details of core %core in repelling group bound %rgb.
SR-052 (info) Core %core in repelling group bound %rgb has %count internal nets with nearby hostile internal nets closer than the b
SR-053 (info) Core %core in repelling group bound %rgb has %count protruding ignored nets that overlap routing blockage (blocka
SR-054 (info) The repelling group bound %rgb contains %count transit cells that are placed within the bound's separation distance o
SR-055 (info) The repelling group bound %rgb contains %count ignored cells that are placed within the bound's separation distance
SR-056 (warning) Skipping type check of split pins of safety_register_group %group because its registers have not been synthesized.
SR-057 (info) Redundancy register %redundancy_register in safety_register_group %group has tap cells {%list_of_tap_cells} place
SR-058 (info) Details of safety_register_group %group.

EXAMPLES

The following is an example of the textual report generated by **report_safety_status**:

```
prompt> report_safety_status

---------- Safety status report ----------
Block: ChipTop
------------------------------------------
safety register rules: 3
safety register groups: 2
safety critical registers: 1
safety critical register with fault priority: 0
fault tolerant registers: 0
tree split buffers: 18
tap cells: 18
voting cells: 8
redundancy registers: 7
---------- Messages ----------
Information: Safety register related object counts for block ChipTop. (SR-000)
Warning: Found safety_register_rule RULE3 without associated safety_register_group. (SR-001)
Error: The design still contains a safety critical register (REG4) that must be replaced by fault tolerant or redundancy logic to abide t
Error: The safety register groups {GRP1 GRP2} are not mutually exclusive. The shared cells are {TAP11 TAP10 TAP9 TAP3 TAP2 T
Error: The safety_register_group GRP2 has 4 redundancy register(s), while 3 are required by safety_register_rule RULE2. (SR-004
Error: Redundancy register rrm1/REG1 in safety_register_group GRP1 does not have tap cells placed adjacent left and right of it. (S
Error: Redundancy register rrm2/REG3 in safety_register_group GRP2 does not have tap cells placed adjacent left and right of it. (S
Error: Redundancy register REG5 in safety_register_group GRP2 does not have tap cells placed adjacent left and right of it. (SR-00
Error: Found a mismatch between the split pin types in safety_register_rule RULE2 and associated split pins {REG5/RN REG5/SN}
Error: Split pin REG5/SN in safety_register_group GRP2 still has to be buffered. (SR-011)
Error: Split pin REG5/RN in safety_register_group GRP2 still has to be buffered. (SR-011)
------- end of Safety status report ------
```

SEE ALSO

check_safety_intent
create_safety_register_rule
create_safety_register_group
report_safety_register_rules
report_safety_register_groups

report_scan_chains

Displays information about the specified scan chains in the current design.

SYNTAX

status **report_scan_chains**
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*scan_chains*]

Data Types

digits integer
scan_chains string or collection

ARGUMENTS

-verbose

Specifies whether the report contains additional data, reporting detailed information about stub_chains associated with the reported scan_chains.

-nosplit

Most of the scan chains information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing software to extract information from the report output.

-significant_digits *digits*

Specifies the number of significant digits (digits to the right of the decimal point) to report. Allowed values are from 0 through 13. The default is 2. Using this option overrides the value set by the **shell.common.report_default_significant_digits** application option.

scan_chains

Specifies the scan_chains to report. This can be either the name/regexp pattern or a collection of multiple scan_chains.

DESCRIPTION

The **report_scan_chains** command displays information about the specified scan chains in the current design.

EXAMPLE

The following command generates a default report for the scan_chains present in current design.

```
prompt> report_scan_chains
****************************************
Report : report_scan_chains
Block  : B1
Version: Q-2019.12-SP2-DEV
Date   : Thu Jan  2 03:07:56 2020
****************************************
Name         Available Bits     Stub Chains
--------------------------------------------------------------------------------
scan1        4                  stub1 stub2 stub3

scan2        4                  stub1 stub2

scan3        0

1
```

SEE ALSO

create_scan_chain(2)
get_scan_chains(2)
remove_scan_chains(2)

report_scan_compression_configuration

Displays options specified by the **set_scan_compression_configuration** command.

SYNTAX

status **report_scan_compression_configuration**

ARGUMENTS

The **report_scan_compression_configuration** command has no arguments.

DESCRIPTION

This command displays options specified by the **set_scan_compression_configuration** command on the current design.

EXAMPLES

The following is an example of a scan compression configuration report:

```
prompt> report_scan_compression_configuration
****************************************
Report : Scan Compression Configuration
Design : top
Version: P-2019.03
Date   : Thu Feb  7 07:08:49 2019
****************************************

----------------------------------------
TEST MODE: all_dft
VIEW     : Specification
----------------------------------------
xtolerance                  High
Inputs                      4
Outputs                     4
Chain Count                 20
Maximum Chain Length        Unspecified
```

```
    Serialize                   disable
    Shift Power Groups          False
    Shift Power Chain Length    Unspecified
    Shift Power Chain Ratio     Unspecified
    Shift Power Disable         Unspecified
    Partition:                  default_partition
```

SEE ALSO

set_scan_compression_configuration(2)

report_scan_configuration

Displays options specified by the **set_scan_configuration** command.

SYNTAX

status **report_scan_configuration**

ARGUMENTS

Thie **report_scan_configuration** command has no arguments.

DESCRIPTION

This command displays the options specified by the **set_scan_configuration** command on the current design. For more information on what is reported, see the EXAMPLES section.

EXAMPLES

The following is an example of a scan configuration report for the global settings (default test mode):

```
prompt> report_scan_configuration
****************************************
Report : Scan configuration
Design : top
Version: P-2019.03
Date   : Thu Feb  7 07:09:33 2019
****************************************

----------------------------------------
TEST MODE: all_dft
VIEW     : Specification
----------------------------------------
Chain count:                    4
Maximum scan chain length:          Unspecified
Clock mixing:                   Mix clocks
Fanout Limit:                   Unspecified
```

```
Insert terminal lockup:          False
Pipelined scan enable:           False
Create PosEdge Chains:           False
Partition:                       default_partition
Lockup type:                     latch
```

SEE ALSO

set_scan_configuration(2)

report_scan_group

Reports all scan groups that were specified using the **set_scan_group** command.

SYNTAX

status **report_scan_group**

ARGUMENTS

The **report_scan_group** command has no arguments.

DESCRIPTION

This command reports all scan groups that were specified using the **set_scan_group** command.

EXAMPLES

The following example show how to use the **report_scan_group** command:

```
prompt> set_scan_group {BLKA/A_reg BLKA/B_reg} MY_GROUP
1
prompt> report_scan_group
****************************************
Report : Scan groups specification
Design : top
Version: P-2019.03
Date   : Thu Feb  7 07:23:13 2019
****************************************

Number of user-defined routed scan groups: 1
--------------------------------------------------------------------------
Scan_group  Cell_#  Instance_name  Serial Routed  Type       Pin
--------------------------------------------------------------------------
MY_GROUP    No scan cells to report
```

SEE ALSO

set_scan_group(2)

report_scan_path

Displays scan paths and scan cells specified by the **set_scan_path** command and displays scan paths inserted by the **insert_dft** command.

SYNTAX

status **report_scan_path**

ARGUMENTS

The **report_scan_path** command has no arguments.

DESCRIPTION

This command displays user-specified scan paths specified by the **set_scan_path** command, as well as the actual scan paths inserted by the **insert_dft** command.

The report contains three different section types.

- VIEW: Specification

  This section contains user-specified scan path information specified with the **set_scan_path** command. This report can contain information on scan path segments, or scan-in, scan-out, and scan-enable signals along with any hookup pin information.

- VIEW: Existing DFT (-chain)

  This section contains summary information about scan paths that currently exist in the design. The summary contains scan-in, scan-out, scan-enable, scan-clock, scan chain name, and scan chain length information in table form, but does not print the ordered scan cell names for each scan chain.

- VIEW: Existing DFT (-cell)

  This section contains detailed information about the scan cell ordering of the scan paths that currently exist in the design.

EXAMPLES

The following is an example of **report_scan_path** in the first format reporting on scan paths in the **spec** view. In this report, the *mychain1* scan path has SI (scan-in), Q (scan-out), and two scan enables SE1 and SE2, which are both hooked up to pin u5/A.

```
prompt> report_scan_path -view spec -chain mychain1 -test_mode mymodeA

****************************************
Report : Scan path
Design : A
Version: 2003.12
Date   : Thu Aug 14 10:46:10 2003
****************************************


========================================
TEST MODE: mymodeA
VIEW    : Specification
========================================
Scan_path        ScanDataIn (h)     ScanDataOut (h)    ScanEnable (h)
---------------  ---------------    ---------------    ---------------
mychain1         SI (-)             Q (-)              SE1 (u5/A)
                                                       SE2 (u5/A)
```

The following is an example of **report_scan_path** in the second format reporting scan paths in the **existing_dft** view. In this report, the *mychain1* scan path is defined in the *mymodeA* test mode, and *mychain2* is defined in the *mymodeB* test mode. The *mychain2* scan chain has SI2 (scan-in), SO (scan-out), and SE2 & SE3 (scan-enable). Clocks MCLK1, MCLK2, and MCLK3 are master clocks for this scan chain, and SCLK is a slave clock.

```
prompt> report_scan_path -view existing_dft -chain all -test_mode all

****************************************
Report : Scan path
Design : A
Version: 2003.12
Date   : Thu Aug 14 10:46:10 2003
****************************************


========================================
TEST MODE: mymodeA
VIEW    : Existing DFT
========================================
Scan_path   Len  ScanDataIn  ScanDataOut ScanEnable  MasterClock SlaveClock
----------  ----- ----------- ----------- ----------- ----------- -----------
mychain1    -    SI         Q          SE1         -          -

========================================
TEST MODE: mymodeB
VIEW    : Existing DFT
========================================
Scan_path   Len  ScanDataIn  ScanDataOut ScanEnable  MasterClock SlaveClock
----------  ----- ----------- ----------- ----------- ----------- -----------
mychain2    5    SI2        SO         SE3         MCLK1       SCLK
                             SE2        MCLK2       -
                             -          MCLK3       -
```

The following is an example of **report_scan_path** in the third format reporting scan cells. In this report, 4 scan cells are reported from the *mychain1* scan chain:

```
prompt> report_scan_path -view existing_dft -cell mychain1 \
```

-test_mode Internal_scan

```
****************************************
Report : Scan path
Design : top
Version: 2003.12
Date   : Thu Aug 14 10:46:29 2003
****************************************

========================================
TEST MODE: Internal_scan
VIEW    : Existing DFT
========================================
Scan_path       Cell_#   Instance_name      Clocks
---------       ------   -------------      ------
mychain1          0       r0/r0/q_reg1/q_reg
                  1       r0/r0/q_reg2/q_reg
                  2       r0/r1/q_reg1/q_reg
                  3       r0/r1/q_reg2/q_reg
```

SEE ALSO

set_scan_path(2)

report_scan_rp_group

Reports Scan relative placement groups defined by **set_scan_rp_group**.

SYNTAX

status **report_scan_rp_group**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **report_scan_rp_group** command reports the Scan relative placement (RP) groups previously defined with the command **set_scan_rp_group**.

The Scan RP groups will be displayed in the tabular format indicating the RP group name, the Scan routing direction (horizontal, vertical), the starting corner (lower left, lower right, upper left and upper right, labeled ll, lr, ul and ur, respectively) and the user max length.

EXAMPLES

The following example reports the Scan relative placement groups set up in the current design.

```
prompt> report_scan_rp_group
****************************************
Report : Scan RP Group Stitching
Design : SUB1
Version: Q-2019.12-SP4
Date   : Fri Apr 20 10:22:32 2020
****************************************

RP Group          Scan Routing     Start Corner     User Length Limit
--------          ------------     ------------     -----------------
RP_1              HORIZONTAL       UR               0
RP_2              VERTICAL         LR               0
```

1

SEE ALSO

set_scan_rp_group(2)
remove_scan_rp_group(2)

report_scan_skew_group

Reports scan skew groups defined by the **set_scan_skew_group** command.

SYNTAX

status **report_scan_skew_group**
scan_skew_group_list

Data Types

scan_skew_group_list list

ARGUMENTS

scan_skew_group_list

When you specify a list of one or more scan skew group names, the command prints a detailed list of the elements in the specified scan skew groups. Wildcards are not supported.

By default, the command reports a summary of all scan skew groups defined, but does not include the individual elements in each scan skew group.

DESCRIPTION

This command displays the user-specified scan skew groups defined by the **set_scan_skew_group** command.

EXAMPLES

To report a summary of the scan skew groups defined, issue the command with no arguments:

```
prompt> report_scan_skew_group

****************************************
Report : Scan Skew Group
Design : top
Version: O-2018.06-SP5-VAL
Date   : Thu Aug 15 00:00:00 2019
```

```
****************************************

----------------------------------------
TEST MODE: all_dft
VIEW    : Specification
----------------------------------------
Scan_skew_group
---------------
Number of user-defined scan skew groups: 3
G1
G2
G3

1
```

To report the detailed list of elements in particular scan skew groups, specify the list of groups to report:

```
prompt> report_scan_skew_group {G1 G2}

****************************************
Report : Scan Skew Group
Design : top
Version: O-2018.06-SP5-VAL
Date   : Thu Aug 15 00:00:00 2019
****************************************

----------------------------------------
TEST MODE: all_dft
VIEW    : Specification
----------------------------------------
Scan_skew_group
---------------
G1          0       Z1P_reg[10]
G1          1       Z1P_reg[11]
G1          2       Z1P_reg[12]

G2          0       Z1P_reg[13]
G2          1       Z1P_reg[14]
G2          2       Z1P_reg[15]

1
```

SEE ALSO

set_scan_skew_group(2)
remove_scan_skew_group(2)

report_scenarios

Prints a brief report of the design's scenarios.

SYNTAX

report_scenarios
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-nosplit]

Data Types

mode_list list
corner_list list
scenario_list list

ARGUMENTS

-modes *mode_list*

Specifies the modes for which to generate the scenario report. Each active scenario of the modes given by the **-modes** option and the corners specified by the **-corners** option will be reported. If this option is not specified, the command reports scenarios of the current mode. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command reports every scenario. It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to generate the scenario report. Each scenario of the modes given by the **-modes** option and the corners specified by the **-corners** option is reported. If this option is not specified, the command reports scenarios of all corners. It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios to report. Each of the given scenarios is reported. It is an error to give this option with the **-modes** or the **-corners** options.

-nosplit

Writes out wide report lines, even if the line length is greater than 80 columns. Use this option to simplify post-processing of the output files or when comparing the output files with the UNIX **diff** command. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

This command prints a brief report for the specified scenarios, or all scenarios if no options are given. The report has one line for each scenario, listing the scenario's name, mode, corner, and analysis flags. If the scenario name was explicitly given by the user (as opposed to being auto-generated), it will be labeled with a "*".

Multicorner-Multimode Support

By default, this command works on all scenarios. To specify different scenarios, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example prints a brief report for all scenarios.

```
prompt> report_scenarios
```

SEE ALSO

all_scenarios(2)
create_scenario(2)
create_scenario(2)
current_scenario(2)
get_scenarios(2)
remove_scenarios(2)

report_secondary_pg_placement_constraints

report all secondary pg constraints

SYNTAX

status **report_secondary_pg_placement_constraints**

ARGUMENTS

There is no argument for this command

DESCRIPTION

The command reports all secondary pg constraints that have been entered and not removed.

This command can be used before or after commit_secondary_pg_placement_constraints. The secondary pg constraints that are reported by this command are not related to whether the constraints have been committed.

EXAMPLES

The following example report all secondary pg constraint

```
prompt> create_secondary_pg_placement_constraints -name pg_cstr0 -supply vdd0 -region {{20 0 } {30 50}}
prompt> report_secondary_pg_placement_constraints
name: pg_cstr0
  supply: vdd0
  margin: 0
  voltage_areas:
  layers:
  regions:
    rectangle: {20 0} {30 50}
```

SEE ALSO

create_secondary_pg_placement_constraints(2)
remove_secondary_pg_placement_constraints(2)
commit_secondary_pg_placement_constraints(2)

report_self_gating_objects

Report the object inclusion options that have been explicitly set for Self-Gating with **set_self_gating_objects**.

SYNTAX

status **report_self_gating_objects**

DESCRIPTION

This command will report the settings that were explicitly set with the **set_self_gating_objects** command. The report will generate a table with the following columns:

- The action, set with **set_self_gating_objects**, the default values or its inherited values when applicable. The possible values are Include, Exclude or Force.
- The tree type option, set with **set_self_gating_objects**, the default values or its inherited values when applicable. The possible values are XOR, NAND, OR and AUTO. If the corresponding action is Exclude, the value will be "-".
- The minimum bitwidth option associated to the tree type value, set with **set_self_gating_options**, the default values or its inherited values when applicable.
- The maximum bitwidth option associated to the tree type value, set with **set_self_gating_options**, the default values or its inherited values when applicable.
- The interaction with clock gating option associated to the tree type value, set with **set_self_gating_options**, the default values or its inherited values when applicable.
- The instance for which the option values are associated.

Notice that the minimum bitwidth, the maximum bitwidth and the interaction with clock gating options are the derived option values for the given intance and tree type. Those option values may be changed using the **set_self_gating_options** command.

EXAMPLES

The following examples will use a design with three hierarchies (hier1, hier2, hier3), below the top level design, named top. The four, hier1, hier2, hier4 and top, have three registers each one.

The following example will use the following setting:

```
prompt> set_self_gating_options -tree_type xor -min 2 -max 4 \
  -interaction collapse
```

```
prompt> set_self_gating_options -tree_type nand -min 4 -max 4 \
  -interaction collapse
prompt> set_self_gating_options -objects {hier1 hier2} -tree_type nand \
  -min 4 -max 8 -interaction none
prompt> set_self_gating_options -objects {hier2} -tree_type xor \
  -min 4 -max 4 -interaction insert
prompt> set_self_gating_objects -include hier1 -tree_type nand
prompt> set_self_gating_objects -force hier2/out_register* -tree_type xor
prompt> set_self_gating_objects -exclude out_register*
```

The example shows the generated report with **report_self_gating_objects**. Since the hier3 hierarchy doesn't have any explicit option set, it is not shown in the report.

```
prompt> report_self_gating_objects

****************************************
Report: Self-Gating Object Settings
Design: top__1
Version: 3.5.0
Date: Thu May 10 12:54:39 2018

****************************************


                    |       Derived from Self-Gating Options        |
  Action   |  Tree Type  | Min Bitwidth | Max Bitwidth | Interaction with CG | Instances
-----------+-------------+--------------+--------------+---------------------+----------
  Exclude  |     -       |      -       |      -       |          -          | out_register[0]
  Exclude  |     -       |      -       |      -       |          -          | out_register[1]
  Exclude  |     -       |      -       |      -       |          -          | out_register[2]
   Force   |    XOR      |      4       |      4       |       Insert        | hier2/out_register[0]
   Force   |    XOR      |      4       |      4       |       Insert        | hier2/out_register[1]
   Force   |    XOR      |      4       |      4       |       Insert        | hier2/out_register[2]
  Include  |    NAND     |      4       |      8       |        None         | hier1
```

SEE ALSO

set_self_gating_options(2)
set_self_gating_objects(2)
report_self_gating_options(2)

report_self_gating_options

Report the options that have been explicitly set for Self-Gating with **set_self_gating_options**.

SYNTAX

status **report_self_gating_options**
[-verbose]

ARGUMENTS

-verbose

By default, the limit of objects to be displayed is 10 for every different setting. With this option all objects of every setting will be displayed.

DESCRIPTION

This command will report the settings that were explicitly set with the **set_self_gating_options** command. For each group of objects with the same explicitly set and derived options, the report will generate a table with the following columns:

- The option name.
- The option value when using XOR combinational logic, set with set_self_gating_objects, the default values or its inherited values when applicable.
- The option value when using NAND combinational logic, set with set_self_gating_objects, the default values or its inherited values when applicable.
- The option value when using OR combinational logic, set with set_self_gating_objects, the default values or its inherited values when applicable.
- The option value when using the automatic combinational logic selection, set with set_self_gating_objects, the default values or its inherited values when applicable.

If an option value has been derived from a parent hierarchy, the option value will have the "(i)" string next to it. If an option value has been obtained from the default values, the option value will have the "(d)" string next to it.

EXAMPLES

The following examples will use a design with three hierarchies (hier1, hier2, hier3), below the top level design, named top. The four, hier1, hier2, hier4 and top, have three registers each one.

The following example will use the following setting:

```
prompt> set_self_gating_options -for_tree_type xor -min 2 -max 4 -interaction collapse
prompt> set_self_gating_options -for_tree_type nand -min 4 -max 4 -interaction collapse
prompt> set_self_gating_options -objects {hier1 hier2} -for_tree_type nand -min 4 -max 8 -interaction none
prompt> set_self_gating_options -objects {hier2} -for_tree_type xor -min 4 -max 4 -interaction insert
```

The example shows the generated report with the **report_self_gating_options**. Since the hier3 hierarchy doesn't have any explicit option set, it is not shown in the report. As the remaining three hierarchies (hier1, hier2 and top) don't have any option set for the **or** and **auto** tree types, their option values are marked as default (d). The hier1 hierarchy doesn't have any option set for the **xor** tree type, but the top hierarchy does. In this case, the options values of this hierarchy and the **xor** tree type is marked as inherited (i).

```
prompt> report_self_gating_options

****************************************
Report: Self-Gating Options
...

****************************************

Objects:
  top
     Options    |   XOR    |   NAND   |    OR    |   AUTO
--------------------+------------+------------+------------+------------
  Minimum Bitwidth |     2    |     4    |   4 (d)  |   4 (d)
  Maximum Bitwidth |     4    |     4    |   8 (d)  |   8 (d)
Interaction with CG|  Collapse  |  Collapse  |  None (d)  |  None (d)

Objects:
  hier1
     Options    |   XOR    |   NAND   |    OR    |   AUTO
--------------------+------------+------------+------------+------------
  Minimum Bitwidth |   2 (i)  |     4    |   4 (d)  |   4 (d)
  Maximum Bitwidth |   4 (i)  |     8    |   8 (d)  |   8 (d)
Interaction with CG|Collapse (i)|    None    |  None (d)  |  None (d)

Objects:
  hier2
     Options    |   XOR    |   NAND   |    OR    |   AUTO
--------------------+------------+------------+------------+------------
  Minimum Bitwidth |     4    |     4    |   4 (d)  |   4 (d)
  Maximum Bitwidth |     4    |     8    |   8 (d)  |   8 (d)
Interaction with CG|   Insert   |    None    |  None (d)  |  None (d)
```

The next example will use the same design, but with the following setting:

```
prompt> set_self_gating_options -for_tree_type xor -min 2 -max 4 -interaction collapse
prompt> set_self_gating_options -for_tree_type nand -min 4 -max 4 -interaction collapse
prompt> set_self_gating_options -objects {hier1 hier2} -for_tree_type nand -min 4 -max 8 -interaction none
prompt> set_self_gating_options -objects {hier2} -for_tree_type xor -min 4 -max 4 -interaction insert
prompt> set_self_gating_options -objects {hier1} -for_tree_type xor -min 4 -max 4 -interaction insert
```

The example shows the generated report with the **report_self_gating_options**, with two objects sharing the same set of options.

```
prompt> report_self_gating_options

****************************************
Report: Self-Gating Options
...

****************************************

Objects:
 top__1
    Options    |   XOR    |   NAND   |    OR    |   AUTO
-------------------+------------+------------+------------+------------
 Minimum Bitwidth |     2    |     4    |   4 (d)  |   4 (d)
 Maximum Bitwidth |     4    |     4    |   8 (d)  |   8 (d)
Interaction with CG|  Collapse  |  Collapse  |  None (d)  |  None (d)

Objects:
 b543
 b876
    Options    |   XOR    |   NAND   |    OR    |   AUTO
-------------------+------------+------------+------------+------------
 Minimum Bitwidth |     4    |     4    |   4 (d)  |   4 (d)
 Maximum Bitwidth |     4    |     8    |   8 (d)  |   8 (d)
Interaction with CG|   Insert   |    None    |  None (d)  |  None (d)
```

SEE ALSO

set_self_gating_options(2)
set_self_gating_objects(2)
report_self_gating_objects(2)

report_serialize_configuration

Displays options specified by the **set_serialize_configuration** command or the **reset_serialize_configuration** command.

SYNTAX

status **report_serialize_configuration**

ARGUMENTS

The **report_serialize_configuration** command has no arguments.

DESCRIPTION

This command displays options specified by the **set_serialize_configuration** command on the current design.

EXAMPLES

The following example shows a serialize configuration report:

```
prompt> report_serialize_configuration
****************************************
Report : Serialize Specification
Design : top
Version: P-2019.03
Date   : Thu Feb  7 07:50:36 2019
****************************************

Inputs                  1
Outputs                 1
Clock                   Unspecified
Update Stage            false
Wide Duty Cycle         false
Exclude Clocks          Unspecified
```

SEE ALSO

set_serialize_configuration(2)

report_shape_patterns

Generates a report of shape_patterns in the current design.

SYNTAX

string **report_shape_patterns**
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*shape_pattern_list*]

Data Types

digits integer
shape_pattern_list collection

ARGUMENTS

-verbose

Specifies whether the report contains additional columns of data, reporting more complete information about the given shape_patterns. By default: name, shape_pattern_type, dimension and displacements are reported. With this option additional attributes are shown.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this application option if you want to override the default. Fixed-precision floating-point arithmetics is used for delay calculation; therefore, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h. The upper bound on a meaningful value of the argument to **-significant_digits** is then FLT_DIG - ceil(log10(fabs(*any_reported_value*))).

-nosplit

Most of the shape_pattern information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing software to extract information from the report output.

shape_pattern_list

Specifies a list of shape_patterns in the given block to be reported. The default is to report all shape_patterns in the given container.

DESCRIPTION

The command displays information about the specified shape_patterns in the block. Attributes such as the dimensions and name of the shape_pattern are displayed.

EXAMPLES

The following example reports shape_patterns of a net:

```
prompt> report_shape_patterns [get_shape_patterns -of_objects \
                               [get_nets VDD]]
Shape_pattern       Type         Dimensions GridSize Displacements
--------------------------------------------------------------------------------
RECT_PATTERN_16_0    rect_pattern  {5x4}    1      {1.5000:eastx2.5000:north}
1
```

The following example displays verbose information for the shape_pattern named RECT_PATTERN_16_0.

```
prompt> report_shape_patterns -verbose RECT_PATTERN_16_0
Shape_pattern       Type         Dimensions GridSize Displacements OccupancyPattern
--------------------------------------------------------------------------------
RECT_PATTERN_16_0    rect_pattern  {5x4}    1      {1.5000:eastx2.5000:north}
                                             1101
1
```

SEE ALSO

create_shape_pattern(2)
get_shape_patterns(2)
remove_shape_patterns(2)
report_shape_patterns(2)
shell.common.report_default_significant_digits(3)

report_shaping_channels

Reports the shaping channels created by the **create_shaping_channel** and **create_shaping_constraint** commands.

SYNTAX

string **report_shaping_channels**
[-header *header_string*]
[-channels *shaping_channels*]
[-designs *designs*]
[-of_objects *shapeable_regions*]
[-types *type_list*]

Data Types

header_string string
shaping_channels list
designs list
shapeable_regions list
type_list list

ARGUMENTS

-header *header_string*

Specifies the string used to replace the text following "Report:" in the report header.

-channels *shaping_channels*

Specifies the list of channels to report. A list of channel names can be obtained with the **get_shaping_channels** command. This option is mutually exclusive with the **-designs**, **-types** and **-of_objects** options.

-designs *designs*

Specifies the list of designs to report. If this option is omitted, the current hierarchical design is reported. This option is mutually exclusive with the **-channels** and **-of_objects** options.

-of_objects *shapeable_regions*

Specify the objects for which to report channels. Valid objects are blocks, block instances, voltage areas, move bounds, and shaping groups. Neighbor-specific channels are included if the neighbor matches one of the specified objects. This option is mutually exclusive with the **-designs** and **-channels** options.

-types *type_list*

Limits the report to only the specified channel types. Valid values are **neighbor** and **object**. By default, all channel types are reported. This option is mutually exclusive with the **-channels** option.

DESCRIPTION

This command prints a report of the shaping channels found in the hierarchy of the current top design or the designs specified with the **-designs** option. Optionally, you can restrict the report to specific types, related objects or channels.

EXAMPLES

The following example reports all shaping channels within the current design and its hierarchically nested blocks visible to **shape_blocks**:

```
prompt> report_shaping_channels
```

The following example reports only neighbor-specific channels.

```
prompt> report_shaping_channels -types neighbor
```

The following example reports only the shaping channels found within the ORCA design.

```
prompt> report_shaping_channels -designs [get_designs ORCA]
```

The following example reports the shaping channels related to movebound MB2:

```
prompt> report_shaping_channels -of_objects [get_bounds MB2]
```

The following example reports the details of the specified shaping channel:

```
prompt> get_shaping_channels
{ORCA/SHAPING_CONSTRAINT_0_child_default_channels/NEIGHBOR_CHANNEL_0}
prompt> report_shaping_channels -channels {ORCA/SHAPING_CONSTRAINT_0_child_default_channels/NEIGHBOR_CHAN
```

SEE ALSO

create_shaping_channel(2)
create_shaping_constraint(2)
get_groups(2)
get_shaping_channels(2)
get_shaping_constraints(2)
report_shaping_constraints(2)
shape_blocks(2)

report_shaping_constraints

Reports the shaping constraints created by the **create_shaping_constraint** command.

SYNTAX

string **report_shaping_constraints**
[-header *header_string*]
[-constraints *shaping_constraints*]
[-designs *designs*]
[-types *type_list*]

Data Types

header_string string
shaping_constraints list
designs list
type_list list

ARGUMENTS

-header *header_string*

Specifies the string used to replace the text following "Report:" in the report header.

-constraints *shaping_constraints*

Specifies the list of constraints to report. This option is mutually exclusive with the **-designs** and **-types** options.

-designs *designs*

Specifies the list of designs to report. If this option is omitted, the current hierarchical design is reported. This option is mutually exclusive with the **-constraints** option.

-types *type_list*

Limits the report to only the specified shaping constraint types. Valid values are: **external_channels**, **child_default_channels**, **boundary_channels**, **target_location**, **reference_region**, **alignment_point**, **parent_object**, **aspect_ratio**, **boundary_area**, **child_default_utilization**, **utilization**, **array_layout**, **allowable_orientation**, and **is_rigid_boundary**. By default, all shaping constraint types are reported. This option is mutually exclusive with the **-constraints** option.

DESCRIPTION

This command prints a report of the shaping constraints found hierarchically within the current top design or the designs specified with the **-designs** option. Optionally, you can restrict the report to specific designs, types, or constraints.

EXAMPLES

The following example reports all shaping constraints within the current design and its hierarchically nested blocks visible to the **shape_blocks** command.

```
prompt> report_shaping_constraints
```

The following example reports only the utilization shaping constraints.

```
prompt> report_shaping_constraints -types utilization
```

The following example reports only the shaping constraints within the ORCA design.

```
prompt> report_shaping_constraints -designs [get_designs ORCA]
```

The following example reports only the shaping constraints related to the movebound MB2.

```
prompt> report_shaping_constraints -constraints [get_shaping_constraints -of_objects MB2]
```

SEE ALSO

create_shaping_channel(2)
create_shaping_constraint(2)
get_groups(2)
get_shaping_channels(2)
get_shaping_constraints(2)
report_shaping_channels(2)
shape_blocks(2)

report_shaping_options

Reports the default shaping options and the current shaping options set by the **set_shaping_options** command.

SYNTAX

int **report_shaping_options**

ARGUMENTS

This command has no arguments.

DESCRIPTION

Reports the default shaping options and the current shaping options set by the **set_shaping_options** command.

EXAMPLES

The following example reports shaping options for the current design.

```
prompt> report_shaping_options
```

SEE ALSO

set_shaping_options(2)

report_shields

Reports statistics for router automatic shield routing.

SYNTAX

status **report_shields**
[-nets *collection_of_nets*]
[-per_layer true | false]
[-output *file_name*]

Data Types

collection_of_nets collection
net_name string
number_of_tracks int
file_name string

ARGUMENTS

-nets *collection_of_nets*

Specifies the nets to report shielding statistics on.

By default, this command works on all shielded nets.

-per_layer true | false

Controls whether to report the shielding ratio statistics for each layer, as well as the overall ratio. If a net is completely shielded, the breakdown for each layer is omitted. The ratio is reported only on layers that have routing for each reported net.

By default (false), this command reports only the overall ratio for each net.

-output *file_name*

Specifies the output file name.

By default, the results are written to a file named report_shields.txt.

DESCRIPTION

The **report_shields** command reports the statistics of a design that was shielded by router automatic shield routing. The statistics are written to the standard output and to a text file. By default, the report file is named report_shields.txt; to specify a different name

for the report file, use the **-output** option.

By default, the command reports on the same-layer shielding on all nets shielded by the **create_shield** command. The report looks the same as the report generated at the end of the **create_shields** command, which reports the overall shielding coverage ratio per net and for the whole design; however, there might be a slight difference in the shield ratios reported by the **create_shields** and **report_shields** commands. This is due to graph connectivity differences between the two commands. When the reported values differ, use the values reported by the **report_shields** command. The difference in reported values is typically less than three percent, but can be up to five percent.

To explicitly specify the nets on which to report shielding, use the **-nets** option.

By default, the command does not report the ratios on a per-layer basis, but rather reports the overall coverage of the entire net.

For coaxial shielding options, the tool automatically detects which options the nets were shielded with and uses those options for reporting on a per-net basis.

Prerequisites

Before you run this command, the you must have created shielding on the design by using the **create_shields** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following reports the overall shielding coverage ratio for the CLK_B1 and CLK_B2 clock nets using the default for all other options. The VSS shielding net is automatically found by the tool from the shielding data. The output shown in the example is the report stored in the text file; the log contains the statistics, but not the report header.

```
prompt> report_shields -nets {CLK_B1 CLK_B2} \
  -output rpt1.txt

************************************
Report : Router Shielding Ratio
      : -nets specified
      : -per_layer false
 ...

************************************

Shielded 99% side-wall of (CLK_B1)
Shielded 97% side-wall of (CLK_B2)
Shielded 2 nets with average ratio as follows.
      1) 98%          (total shield ratio/number of shielded nets)
      2) 97%          (total shield length/total shielded net length)

************************************
```

The following example reports the shielding ratio per layer for the CLK_B5 net, which has coaxial shielding above and below the shielded net segment layers and has no signal routing resources between the coaxial shielding segments. The coaxial options and VSS shielding net are automatically found by the tool from the shielding data.

```
prompt> report_shields -nets {CLK_B5} -per_layer true \
```

```
  -output rpt2.txt

***********************************
Report : Router Shielding Ratio
      : -nets specified
      : -per_layer true
 ...

***********************************

Shielded 90% side-wall of (CLK_B5); 297 coaxial shielding wires added
      Layer          M3 :     95%
      Layer          M4 :     90%
      Layer          M5 :     85%
Shielded 1 nets with average ratio as follows.
      1) 90.00%          (total shield ratio/number of shielded nets)
      2) 91.00%          (total shield length/total shielded net length)

***********************************
```

The following example uses all defaults. It reports all shielded nets using the saved shielding options.

```
prompt> report_shields -output rpt3.txt

***********************************
Report : Router Shielding Ratio
      : -nets all shielded
      : -per_layer false
 ...

***********************************

Shielded 99% side-wall of (CLK_B1)
Shielded 97% side-wall of (CLK_B2)
Shielded 90% side-wall of (CLK_B5); 297 coaxial shielding wires added
Shielded 2 nets with average ratio as follows.
      1) 95.33%        (total shield ratio/number of shielded nets)
      2) 96.25%        (total shield length/total shielded net length)

***********************************
```

SEE ALSO

create_shields(2)

report_si_calculation

Displays the SI delta delay calculations for an entire stage

SYNTAX

status **report_si_calculation**
[-mode *mode*]
[-corner *corner*]
[-scenario *scenario*]
[-min]
[-max]
[-rise]
[-fall]
[-clock *clock*]
[-source_rise]
[-source_fall]
[-significant_digits *digits*]
[*victim_driver_list*]

Data Types

mode	collection
corner	collection
scenario	collection
clock	collection
digits	int
victim_driver_list	list

ARGUMENTS

-mode *mode*

Specifies the mode to use for reporting. The scenario (if any) of the mode specified by the **-mode** option and the corner specified by the **-corner** option will be used. If none of the **-mode**, **-corner**, or **-scenario** options are specified, the command generates a report based on the current scenario. It is an error to give this option with the **-scenario** option.

-corner *corner*

Specifies the corner to use for reporting. The scenario (if any) of the corner specified by the **-corner** option and the mode specified by the **-mode** option will be used. If none of the **-corner**, **-mode**, or **-scenario** options are specified, the command generates a report based on the current scenario. It is an error to give this option with the **-scenario** option.

-scenario *scenario*

Specifies the scenario to use for reporting. It is an error to give this option with the **-mode** or the **-corner** options. If none of the **-**

corner, **-mode**, or **-scenario** options are specified, the command generates a report based on the current scenario.

-max

Reports the calculations for max (late) delta delays. If neither of the **-max** or **-min** options are specified, max delta delays will be reported. If both of these options are specified, each type of calculation will be reported separately.

-min

Reports the calculations for min (early) delta delays. If neither of the **-max** or **-min** options are specified, max delta delays will be reported. If both of these options are specified, each type of calculation will be reported separately.

-rise

Reports the calculations for rising victim driver transitions. If neither of the **-rise** or **-fall** options are specified, rise delta delays will be reported. If both of these options are specified, each type of calculation will be reported separately.

-fall

Reports the calculations for falling victim driver transitions. If neither of the **-rise** or **-fall** options are specified, rise delta delays will be reported. If both of these options are specified, each type of calculation will be reported separately.

-clock *clock*

Uses existing transition data for the specified clock, and clock timing derate values (if any), to calculate delta delays. If this option is not specified, the data transitions and derate values will be used. If the cell or net arcs being reported are not part of the clock's network, a warning will be issued. If the clock does not belong to the mode being reported, the command will attempt to use a clock of the same name from the correct mode.

-source_rise

Specifies that the rising clock edge should be used to determine the transition time at the source node of the timing arc. It is an error to specify **-source_rise** or **-source_fall** without the *-clock* option. If neither **-source_rise** or **-source_fall** is specified with the *-clock* option, the clock source rise/fall will be the same as the rise/fall at the driver. That is, the clock network will be assumed to be non-inverting. If both **-source_rise** or **-source_fall** are specified, each type of calculation will be reported separately.

-source_fall

Specifies that the falling clock edge should be used to determine the transition time at the source node of the timing arc. It is an error to specify **-source_rise** or **-source_fall** without the *-clock* option. If neither **-source_rise** or **-source_fall** is specified with the *-clock* option, the clock source rise/fall will be the same as the rise/fall at the driver. That is, the clock network will be assumed to be non-inverting. If both **-source_rise** or **-source_fall** are specified, each type of calculation will be reported separately.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 through 13. The default is 2. Using this option overrides the value set by the **shell.common.report_default_significant_digits** application option.

victim_driver_list

Reports the victim stages driven by the specified pins or ports.

DESCRIPTION

The **report_si_calculation** command creates a detailed report of the delta delay and slew calculations for a stage, which is the combination of a driving cell or port and the driven net, with its interconnect parasitics and loads.

The report includes a summary of the relevant PVT and delay calculation parameters, descriptions of the delay calculations of the victim stage's driver, a description of the net's parasitic RC network, the transition time of each aggressor net and its contribution to the delta delay, and the delay, delta delay, slew, and slew degradation of each load pin.

The **-min**, **-max**, **-rise**, **-fall**, **-clock**, *-source_rise*, and *-source_fall* options control which types of delta delays are reported. Note that each specified min/max/rise/fall/source_rise/source_fall combination will be reported separately. If multiple drivers are specified, each one will be reported separately. It is an error if the specified pin or port is not a driver. The reporting will be based on the current mode and corner. Note that the data calculations will be done by default, even if the stage is part of a clock network.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenario** option or the **-corner** and **-mode** options.

EXAMPLES

The following command generates a signal integrity report for the current design.

```
prompt> report_si_calculation
```

SEE ALSO

report_delay_calculation(2)
report_stage(2)

report_signal_em

Reports signal electromigration results for the design or specified nets.

SYNTAX

report_signal_em
[-verbose]
[-violated]
[-significant_digits *digits*]
[-nets *net_list*]

Data Types

digits integer
net_list list

ARGUMENTS

-verbose

Reports detailed electromigration analysis information for the nets in the design or violating nets. If **-violated** is specified, it only reports violating nets. Otherwise, it reports all the nets.

-violated

Reports the detailed electromigration analysis information for the violating nets of the design. Works with **-verbose**.

-significant_digits *digits*

Specifies the number of digits after the decimal point to be displayed for values in the generated report. The default value is **2**.

-nets *net_list*

Specifies the list of net names for analysis.

DESCRIPTION

The **report_signal_em** command performs signal (clock and data) electromigration analysis on the current design and generates a report. It displays signal EM QoR for the design or specified nets.

The command performs electromigration analysis of each net by calculating the current on every segment of the net and comparing it with the current limits defined by constraints.

By default, **report_signal_em** analyzes the whole design, including clock nets and signal nets. The net list could be specified for certain group of nets. For example, command **get_nets** could be used to filter out clock nets, **[get_nets -hier * -filter "net_type!=Clock"]**.

Before running **report_signal_em**, make sure the design has signal electromigration constraints. You can load an ITF-EM file into an existing design library.

Also, be sure that the switching activity has been defined for all boundary nets.

This command uses information from current mode and corner.

EXAMPLES

The following example runs signal electromigration analysis and includes detailed violating nets information in the reporting.

```
prompt> report_signal_em -verbose -violated
```

SEE ALSO

set_switching_activity(2)
read_signal_em_constraints(2)

report_signal_io_constraints

Reports signal I/O constraints of the specified I/O guides.

SYNTAX

int **report_signal_io_constraints**
[-io_guide_list *io_guide*]
[-significant_digits *digits*]

Data Types

io_guide list
digits int

ARGUMENTS

-io_guide_list

Specifies the name or collection of I/O guides to report. If no I/O guides are specified, signal constraints of all I/O guides are reported.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13; the default is determined by the **shell.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default.

DESCRIPTION

This command reports signal constraints existed on I/O guides. The command lists all pad instances described as part of the signal constraints. For each such pad, the command writes a location value as the distance measured from the starting point of the I/O guide.

EXAMPLES

The following example reports signal constraints of an I/O guide named "north".

prompt> **report_signal_io_constraints -io_guide_list north**

The following example reports signal constraints of all I/O guides.

```
prompt> report_signal_io_constraints
```

SEE ALSO

place_io(2)
remove_signal_io_constraints(2)
set_signal_io_constraints(2)

report_site_defs

Displays information about the specified site definitions in the technology data of the of current or specified library.

SYNTAX

status **report_site_defs**
[-library *library*]
[-tech *tech_object*]
[-nosplit]
[-significant_digits *digits*]
site_defs

Data Types

library string or collection
tech_object collection
digits integer
site_defs string or collection

ARGUMENTS

-library *library*

Site definition will be searched in the technology of this library. Default is current library's tech. This is mutually exclusive with -tech option.

-tech *tech_object*

Site definition will be searched in this technology. Default is current library's tech. This is mutually exclusive with -library option.

-nosplit

Most of the site_def information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

-significant_digits *digits*

Specifies the number of significant digits (digits to the right of the decimal point) to report. Allowed values are from 0 through 13. The default is 2. Using this option overrides the value set by the **shell.common.report_default_significant_digits** application option.

site_defs

Specifies the site_defs to report. This can be either the name/regexp pattern or a collection of multiple site_defs.

DESCRIPTION

The **report_site_defs** command displays information about the specified site_defs in the tech of current or specified library. The command reports the name, width, height, type, is_default, tech, symmetry and legal_orientations attributes of the site_defs.

EXAMPLE

The following command generates a default report for the site_defs in the tech of library r4000_lib.

```
prompt> report_site_defs -library r4000_lib

****************************************
Report : report_site_defs
 ...

****************************************
Site_def         Width   Height  Type  Is_default  Tech
  Symmetry  Legal_orientations
--------------------------------------------------------------------------------
lib_site_def_5    5.00    10.00   core  --        r4000_tech
  --       R0
lib_site_def_6    10.00   15.00   pad   --        r4000_tech
  Y        R0 MY
lib_site_def_7    15.00   20.00   core  --        r4000_tech
  X Y      R0 R180 MX MY
lib_site_def_8    20.00   25.00   pad   --        r4000_tech
  X Y R90   R0 R90 R180 R270 MX MXR90 MY MYR90
sd1              5.00    10.00   core  default    r4000_tech
  --       R0
1
```

SEE ALSO

create_site_defs(2)
get_site_defs(2)
remove_site_defs(2)

report_size_only

Reports size_only information for design objects.

SYNTAX

string **report_size_only**
  [-all]
  [*cell_list*]

list *cell_list*

ARGUMENTS

-all

Reports for all size_only cells, hierarchical pins, and flat nets.

cell_list

Lists cells to report. If you do not specify this or the **-all** option, a summary report will be printed, listing the total number of size_only cells, hierarchical pins, and flat nets.

DESCRIPTION

Displays size_only information for cells, hierarhical pins, and nets in the current design. For each size_only object, the reasons for the size_only status will be listed, along with the relevant mode(s) or corner(s).

EXAMPLES

The following example shows a summary report:

```
prompt> report_size_only

Report : size_only
0 of 251 instances are size_only
4 of 163 hierarchical pins are size_only
4 of 275 flat nets are size_only
1
```

The following example shows a report for one cell that has had **set_size_only** applied to it:

```
prompt> report_size_only cell_3/ABO_CELL_19

Report : size_only

Instance cell_3/ABO_CELL_19 (ND2D1BWP)
  Mode/corner independent:  user
1 of 1 instances are size_only
```

The following example lists all size_only objects and the reasons for the size_only status:

```
prompt> report_size_only -all

Report : size_only

Instance cell_3/ABO_CELL_19 (ND2D1BWP)
  Mode/corner independent:  user

Flat net data_out[4]
  Mode default:  exception

Flat net data_out[2]
  Mode default:  exception

Flat net data_out[1]
  Mode default:  exception

Flat net cell_1/cell_8/ABO_NET_21
  Mode default:  exception

Hierarchical pin cell_2/res[1]
  Mode default:  exception

Hierarchical pin cell_2/res[0]
  Mode default:  exception

Hierarchical pin cell_1/cell_8/cell_4/res_0
  Mode default:  exception

Hierarchical pin cell_1/cell_8/cell_4/res_1
  Mode default:  exception
1 of 251 instances are size_only
4 of 163 hierarchical pins are size_only
4 of 275 flat nets are size_only
1
```

SEE ALSO

set_size_only(2)
report_cell(2)

report_skew_macros

Shows the settings that were specified in **set_skew_macros** command.

SYNTAX

int **report_skew_macros**
[-bank_name *name*]

ARGUMENTS

-bank_name *name*

Specifies the bank whose **set_skew_macros** settings should be shown. If no bank is specified, the settings for all banks are shown.

DESCRIPTION

The **report_skew_macros** command reports the specification of **set_skew_macros** commands. If bank_name is not specified, it shows all the specifications of all banks.

EXAMPLES

This example reports the bank 'tlb_tag' set_skew_macros specification.

prompt> **report_skew_macros -bank_name tlb_tag**
```
****************************************
Report : skew_macros
Design : test_core
Version: 3.5.0
Date   : Fri May 11 00:48:06 2018
****************************************


BankName     Improve_Side     Macros

-----------------------------------------------------------------------------------------
tlb_tag      output       u_tlb_tag_bank0
```

```
                    u_tlb_tag_bank1
                    u_tlb_tag_bank2
                    u_tlb_tag_bank3
---------------------------------------------------------------------------------------------
```

SEE ALSO

set_skew_macros(2)
remove_skew_macros(2)

report_stage

Displays the Arnoldi and/or Elmore delay calculations for an entire stage.

SYNTAX

status **report_stage**
[-mode *mode*]
[-corner *corner*]
[-scenario *scenario*]
[-min]
[-max]
[-rise]
[-fall]
[-clock *clock*]
[-source_rise]
[-source_fall]
[-significant_digits *digits*]
[-startpoint_models]
[*driver_list*]

Data Types

mode collection
corner collection
scenario collection
clock collection
digits int
driver_list list

ARGUMENTS

-mode *mode*

Specifies the mode to be used for reporting. The scenario (if any) of the mode specified by the **-mode** option and the corner specified by the **-corner** option will be used. If none of the **-mode**, **-corner**, or **-scenario** options are specified, the command reports using data from the current scenario. It is an error to specify this option with the **-scenario** option.

-corner *corner*

Specifies the corner to be used for reporting. The scenario (if any) of the corner specified by the **-corner** option and the mode specified by the **-mode** option will be used. If none of the **-corner**, **-mode**, or **-scenario** options are specified, the command reports using data from the current scenario. It is an error to specify this option with the **-scenario** option.

-scenario *scenario*

Specifies the scenario to be used for reporting. It is an error to specify this option with the **-mode** or the **-corner** options. If none of the **-corner**, **-mode**, or **-scenario** options are specified, the command reports using data from the current scenario.

-min

Reports the calculations for min (early) delays. If neither of the **-max** or **-min** options are specified, max delays will be reported. If both of these options are specified, each type of calculation will be reported separately.

-max

Reports the calculations for max (late) delays. If neither of the **-max** or **-min** options are specified, max delays will be reported. If both of these options are specified, each type of calculation will be reported separately.

-rise

Reports the calculations for rising driver transitions. If neither of the **-rise** or **-fall** options are specified, rise delays will be reported. If both of these options are specified, each type of calculation will be reported separately.

-fall

Reports the calculations for falling driver transitions. If neither of the **-rise** or **-fall** options are specified, rise delays will be reported. If both of these options are specified, each type of calculation will be reported separately.

-clock *clock*

If this option is specified, existing transition data for the specified clock, and clock timing derate values (if any), will be used to calculate delays. If this option is not specified, the data transitions and derate values will be used. If the cell or net arcs being reported are not part of the clock's network, a warning will be issued. If the clock does not belong to the mode being reported, the command will attempt to use a clock of the same name from the correct mode.

-source_rise

Specifies that the rising edge should be used to determine the transition time at the source node of the timing arc. This option must be specified together with the **-clock** option, otherwise the tool issues an error. If neither the **-source_rise** or **-source_fall** option is specified with the **-clock** option, the clock source rise/fall will be the same as the rise/fall at the driver. That is, the clock network will be assumed to be non-inverting. If both the **-source_rise** or **-source_fall** options are specified, each type of calculation is reported separately.

-source_fall

Specifies that the falling edge should be used to determine the transition time at the source node of the timing arc. This option must be specified together with the **-clock** option, otherwise the tool issues an error. If neither the **-source_rise** or **-source_fall** option is specified with the **-clock** option, the clock source rise/fall will be the same as the rise/fall at the driver. That is, the clock network will be assumed to be non-inverting. If both the **-source_rise** or **-source_fall** options are specified, each type of calculation is reported separately.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 through 13. The default is 2. Using this option overrides the value set by the **shell.common.report_default_significant_digits** application option.

-startpoint_models

Uses the existing driver models for timing startpoints.

driver_list

Reports the stages driven by the specified pins or ports.

DESCRIPTION

The **report_stage** command provides a detailed report of the delay and slew calculations for a stage, which is the combination of a driving cell or port and the driven net, with its interconnect parasitics and loads.

The report will include a summary of the relevant PVT and delay calculation parameters, descriptions of the delay calculations of each driver fanin cell arc, which of those arcs is actually used to drive the net, a description of the net's parasitic RC network, and the delay, slew, and slew degradation of each load pin.

If the tool has Arnoldi delay analysis activated, and the stage can support Arnoldi analysis, a description of the Arnoldi calculations will be reported, along with the equivalent Elmore calculation for comparison. Otherwise, only Elmore calculations will be reported.

The **-min**, **-max**, **-rise**, **-fall**, **-clock**, **-source_rise**, and **-source_fall** options control which types of delays are reported. Note that each specified min/max/rise/fall/source_rise/source_fall combination will be reported separately. If multiple drivers are specified, each one will be reported separately. It is an error of a specified pin or port is not a driver. The reporting will be based on the specified scenario. Note that the data calculations will be done by default, even if the stage is part of a clock network.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenario** option or the **-corner** and **-mode** options.

SEE ALSO

report_delay_calculation(2)

report_starrc_in_design

Report the setting of StarRC in design

SYNTAX

string **report_starrc_in_desgin**

DESCRIPTION

Report the settings used in StarRC in Design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLE

The following command reports the setting for StarRC in design:

```
prompt> report_starrc_in_design
 config = ./starRC/star.config
1
```

SEE ALSO

set_starrc_in_design(2)

report_starrc_options

Report the setting of StarRC

SYNTAX

string **report_starrc_options**

DESCRIPTION

Report the settings used in StarRC which is used in PT ECO fusion.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLE

The following command reports the setting for StarRC:

```
prompt> report_starrc_options
 config = ./starRC/star.config
1
```

SEE ALSO

set_starrc_options(2)

report_stub_chains

Displays information about the specified stub chains in the current design.

SYNTAX

status **report_stub_chains**
[-length]
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*stub_chains*]

Data Types

digits integer
stub_chains string or collection

ARGUMENTS

-length

Specifies whether the report contains manhattan wire length of the stub chains to be reported.

-nosplit

Most of the stub chains information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing software to extract information from the report output.

-significant_digits *digits*

Specifies the number of significant digits (digits to the right of the decimal point) to report. Allowed values are from 0 through 13. The default is 2. Using this option overrides the value set by the **shell.common.report_default_significant_digits** application option.

stub_chains

Specifies the stub_chains to report. This can be either the name/regexp pattern or a collection of multiple stub_chains.

-verbose

Specifies whether the report contains additional physical data, reporting more complete information about the given stub chain.

DESCRIPTION

The **report_stub_chains** command displays information about the specified stub chains in the current design.

EXAMPLE

The following command generates a default report for the stub_chains present in current design.

```
prompt> report_stub_chains
****************************************
Report : report_stub_chains
Block  : top
Version: Q-2019.12-SP2-DEV
Date   : Fri Jan 17 05:51:24 2020
****************************************
STUBCHAIN   : 1
START       : SIDEF0
STOP        : SODEF0
PARTITION   : clk_50_56_DEFAULT_POWER_DOMAIN_VDD_I_default_partition
STATUS      : VALIDATED
--------------------------------------------------------------------------------
rd_reg ( IN SI ) ( OUT Q )
re_reg ( IN SI ) ( OUT Q )
rh_reg ( IN SI ) ( OUT Q )
rg_reg ( IN SI ) ( OUT Q )
rf_reg ( IN SI ) ( OUT Q )

STUBCHAIN   : 2
START       : SIDEF1
STOP        : SODEF1
PARTITION   : clk_50_56_DEFAULT_POWER_DOMAIN_VDD_I_default_partition
STATUS      : VALIDATED
--------------------------------------------------------------------------------
ri_reg ( IN SI ) ( OUT Q )
rj_reg ( IN SI ) ( OUT Q )
rc_reg ( IN SI ) ( OUT Q )
ra_reg ( IN SI ) ( OUT Q )
rb_reg ( IN SI ) ( OUT Q )

1


prompt> fc_shell> report_stub_chains -length
****************************************
Report : report_stub_chains
Block  : top
Version: Q-2019.12-SP2-DEV
Date   : Fri Jan 17 05:58:27 2020
****************************************
```

```
STUBCHAIN  : 1
START      : SIDEF0
STOP       : SODEF0
PARTITION  : clk_50_56_DEFAULT_POWER_DOMAIN_VDD_I_default_partition
STATUS     : VALIDATED
LENGTH     : 218.059
--------------------------------------------------------------------------------
rd_reg ( IN SI ) ( OUT Q )
re_reg ( IN SI ) ( OUT Q )
rh_reg ( IN SI ) ( OUT Q )
rg_reg ( IN SI ) ( OUT Q )
rf_reg ( IN SI ) ( OUT Q )

STUBCHAIN  : 2
START      : SIDEF1
STOP       : SODEF1
PARTITION  : clk_50_56_DEFAULT_POWER_DOMAIN_VDD_I_default_partition
STATUS     : VALIDATED
LENGTH     : 320.075
--------------------------------------------------------------------------------
ri_reg ( IN SI ) ( OUT Q )
rj_reg ( IN SI ) ( OUT Q )
rc_reg ( IN SI ) ( OUT Q )
ra_reg ( IN SI ) ( OUT Q )
rb_reg ( IN SI ) ( OUT Q )

1
```

SEE ALSO

create_stub_chain(2)
get_stub_chains(2)
remove_stub_chains(2)

report_supernet_exceptions

Reports the supernet transparent pins and disabled cells in the current design.

SYNTAX

status **report_supernet_exceptions**
[-nosplit]

ARGUMENTS

-nosplit

Does not split lines if column overflows.

DESCRIPTION

This command reports all the pins which are marked as transparent for supernets and also reports the disabled cells in the current design, as specified by the **set_supernet_exceptions -pins** and **set_supernet_exceptions -disable_cells** command.

EXAMPLES

The following example reports the supernet transparent pins and disabled cells of the design *r4000*:

```
prompt> report_supernet_exceptions

Instance          Supernet Transparent Pins
--------------------------------------------------------------------------------
cntrl/U166              { A1 Z }
alu/r281/U449           { A1 ZN }
U320            { A1 ZN A2 ZN }
--------------------------------------------------------------------------------
Supernet Disabled cells
--------------------------------------------------------------------------------
U1
1
```

SEE ALSO

remove_supernet_exceptions(2)
set_supernet_exceptions(2)

report_supply_net

Reports information about the specified supply nets.

SYNTAX

status **report_supply_nets**
[-nosplit]
[*supply_nets*]

Data Types

supply_nets list

ARGUMENTS

-nosplit

Does not split lines if column overflows.

supply_nets

Specifies the supply nets be reported. The command fails if it does not find any of the specified supply nets. If this option is not specified, all the supply nets in the current scope are reported.

DESCRIPTION

The **report_supply_nets** command reports detailed information about existing supply nets. If you specify the supply nets, only those supply nets are reported; otherwise all supply nets in the current scope are reported.

Until commit_upf command is executed and power intent is finalized, supply nets of subblocks are not propagated and will not be included in the report.

The report for a supply net includes the following information: its full name, its maximum and minimum voltages if set using the **set_voltage** command, its resolve type, and the various supply states. The names of the power domains in which the supply net is available are also printed. The name of the power domain is suffixed with an asterisk (*) if the net is set as the domain supply net for that domain.

If the voltage of the supply net is not set, a '--' is printed, indicating that the voltage is undefined.

This command returns 1 on success, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports information about all the supply nets in the current scope.

```
prompt> report_supply_nets
****************************************
Report : Supply Net
Design : top
Corner : default
 ...

****************************************
------------------------------------------------------
Supply Net           : vdd
Current Scope        : top (top level)
Operating Voltage    : -- Best Case --   -- Worst Case --
                     0.80            1.20
Supply States        : N/A
Available Power Domain : pd
Supply ports         : vdd, U11/vdd
Resolve Type         : unresolved
------------------------------------------------------
 1
```

SEE ALSO

create_supply_net(2)
create_power_domain(2)
connect_supply_net(2)
set_domain_supply_net(2)
get_supply_nets(2)

report_supply_nets

Reports information about the specified supply nets.

SYNTAX

status **report_supply_nets**
[-nosplit]
[*supply_nets*]

Data Types

supply_nets list

ARGUMENTS

-nosplit

Does not split lines if column overflows.

supply_nets

Specifies the supply nets be reported. The command fails if it does not find any of the specified supply nets. If this option is not specified, all the supply nets in the current scope are reported.

DESCRIPTION

The **report_supply_nets** command reports detailed information about existing supply nets. If you specify the supply nets, only those supply nets are reported; otherwise all supply nets in the current scope are reported.

Until commit_upf command is executed and power intent is finalized, supply nets of subblocks are not propagated and will not be included in the report.

The report for a supply net includes the following information: its full name, its maximum and minimum voltages if set using the **set_voltage** command, its resolve type, and the various supply states. The names of the power domains in which the supply net is available are also printed. The name of the power domain is suffixed with an asterisk (*) if the net is set as the domain supply net for that domain.

If the voltage of the supply net is not set, a '--' is printed, indicating that the voltage is undefined.

This command returns 1 on success, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports information about all the supply nets in the current scope.

```
prompt> report_supply_nets
****************************************
Report : Supply Net
Design : top
Corner : default
 ...

****************************************
------------------------------------------------------
Supply Net           : vdd
Current Scope        : top (top level)
Operating Voltage    : -- Best Case --   -- Worst Case --
                     0.80            1.20
Supply States        : N/A
Available Power Domain : pd
Supply ports         : vdd, U11/vdd
Resolve Type         : unresolved
------------------------------------------------------
 1
```

SEE ALSO

create_supply_net(2)
create_power_domain(2)
connect_supply_net(2)
set_domain_supply_net(2)
get_supply_nets(2)

report_supply_ports

Reports information about the specified supply ports.

SYNTAX

status **report_supply_ports**
[-nosplit]
[*supply_ports*]

Data Types

supply_ports list

ARGUMENTS

-nosplit

Does not split lines if column overflows.

supply_ports

Specifies the supply ports to be reported. The command fails if it does not find any of the specified supply ports. If this option is not specified, the tool reports on all supply ports in the current scope.

DESCRIPTION

The **report_supply_ports** command reports detailed information about existing supply ports. If *supply_ports* is not specified, the tool reports on all supply ports within the current scope.

Until commit_upf command is executed and power intent is finalized, supply ports of subblocks are not propagated and will not be included in the report.

The report includes the full name of the supply port, its direction, and the supply nets to which it is connected. Supply ports of power switch strategy can be reported by specifying the full name of the power switch and the supply port name as follows:

report_supply_ports *switch_full_path_name*/*supply_port_name*

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports on all supply ports in the current scope:

```
prompt> report_supply_ports
****************************************
Report : Supply Port
Design : top
 ...

****************************************
----------------------------------------
Supply Port       : vdd
Current Scope     : box1 (top level)
Direction         : in
HighConn Supply Net : N/A
LowConn Supply Net  : vdd
----------------------------------------
Supply Port       : vss
Current Scope     : box1 (top level)
Direction         : in
HighConn Supply Net : N/A
LowConn Supply Net  : vss
----------------------------------------
 1
```

SEE ALSO

create_supply_port(2)
get_supply_ports(2)

report_supply_sets

Report the specified supply sets in the current scope.

SYNTAX

status **report_supply_sets**
  [-nosplit]
  [*supply_set_name*]

Data Types

supply_set_name string list

ARGUMENTS

-nosplit

Does not split lines if column overflows.

supply_set_name

Specifies the supply sets that are to be reported. If the supply set does not exist in the current scope, the command fails.

DESCRIPTION

The **report_supply_sets** command reports detailed information about the supply sets in the current scope.

Until commit_upf command is executed and power intent is finalized, supply sets of subblocks are not propagated and will not be included in the report.

The report for a supply set includes the following information: its full name, the scope in which it was created. If the actual supply nets have been associated with the power and ground functionalities of the specified supply sets, the pairs of function, supply_net are also printed.

This command returns 1 on success, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example reports information about the supply set primary_sset in the current scope.

```
prompt> create_supply_set primary_sset \
        -function {power VDD} \
        -function {ground VSS} \
primary_sset

prompt> report_supply_sets primary_sset

 ****************************************
 Report : supply_set
 Design : top
 ...

****************************************

 --------------------------------------------------------------------------------

 Supply Set          : primary_sset
 Current Scope       : top

 Function            : -- Function --   -- Supply Net --
                       power            VDD
                       ground           VSS
 --------------------------------------------------------------------------------
1
```

SEE ALSO

create_supply_set(2)
create_power_domain(2)

report_switching_activity

Reports statistics about switching activity information in the current design or instance.

SYNTAX

int **report_switching_activity**
[-verbose]
[-nosplit]
[-cells *cell_list*]
[-hierarchy]
[-switching_activity_types *switching_type_list*]
[-scenarios *scenario_list*]
[-modes *mode_list*]
[-corners *corner_list*]
[-coverage]
[-list_low_activity]
[-toggle_rate_limit *limit*]
[-essential]

Data Types

cell_list list
switching_type_list list
scenario_list list
mode_list list
corner_list list
limit float

ARGUMENTS

-verbose

It matches the current default output of the command.

-nosplit

Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line starting in the correct column.

-cells *cells_list*

Indicates that switching activity information is to be reported only for the specified list of cells.

-hierarchy

When the command is used with this option, the report generated includes information about subblocks in the design.

-switching_activity_types *switching_type_list*

When used the report includes only those nets whose activity source matches the ones specified by the user. The allowed switching types are 'annotated', 'propagated', 'simulated', 'sta' and 'default'.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Output of the command is generated separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command reports for the current scenario in the design.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering on modes occurs.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If this option is not specified, no filtering occurs on corners.

-coverage

Reports the nets having toggle rate greater than or equal to the value specified by the **-toggle_rate_limit** option. Coverage is defined as the percentage of nets with toggle rate more than or equal to the toggle rate limit. This report can be used to verify that switching activities from simulation or propagation are properly exercising the design. Combined with the **-hierarchy** option, it can be used to make sure that each block in the design is properly exercised. This option can not be used with any of the **-list_low_activity**, **-switching_activity_types** or **-essential** options.

-list_low_activity

Reports the list of nets in the design having toggle rate less than or equal to the toggle rate limit, which is specified using the **-toggle_rate_limit** option. This report can be used to help debug which nets are not being exercised by the simulation testbench. This option cannot be used with the **-hierarchy** or **-coverage** options.

-toggle_rate_limit *limit*

Sets the toggle rate limit, which is used by the **-coverage** report and the **-list_low_activity** report. Toggle rate limit represents the maximum toggle rate considered to be low activity. The default limit is 0. This option can only be exercised when either of **-coverage** or **-list_low_activity** options has been specified.

-essential

Reports the activity on the essential activity points. Following are the essential activity points : primary input pins, blackbox output pins, edge-sensitive sequential output pins, level-sensitive sequential output pins, logical cycle and user-defined output pins. This option can not be used with the **-coverage** option.

DESCRIPTION

The command has been deprecated in 18.06-SP4 release and will be removed in future releases. Please use **report_activity** instead.

Reports the switching activities on the nets in the current design or instance.

The default report generated by the **report switching_activity** command gives an overview of switching activity information in the current instance. The report lists the number of nets with switching activity that is either user annotated, simulated (read from a file), propagated, sta (derived from clock waveforms) or default.

Switching activity can be annotated by using the **set_switching_activity** and **read_saif** commands.

EXAMPLES

The following examples report switching activity for the current design.

```
prompt> report_switching_activity
****************************************
Report : switching_activity
Design : ChipTop
Version: I-2014.03
Date   : Wed Apr  9 04:30:04 2014
****************************************
Mode: default
Corner: default
Time unit: 1.00ns

Switching Activity Overview Statistics for 'ChipTop'
--------------------------------------------------------------------------------

Object Type    annotated  propagated  simulated  sta  default  Total
--------------------------------------------------------------------------------

Nets              184         0         0    0    7460   7644
--------------------------------------------------------------------------------

Nets Driven by
--------------------------------------------------------------------------------

Primary Input       1         0         0    0     124    125
Memory              0         0         0    0       0      0
Black Box           0         0         0    0       0      0
Clock Gate          0         0         0    0      64     64
Tri-State           0         0         0    0       0      0
Sequential        183         0         0    0     859   1042
Combinational       0         0         0    0    6413   6413

--------------------------------------------------------------------------------
1
```

In the following example, the **report_switching_activity** command is used with the *-switching_activity_type* option to find the nets in the design that have switching activity information of types 'default' and 'annotated'.

```
prompt> report_switching_activity \
  -switching_activity_types {annotated default}
****************************************
Report : switching_activity
Design : ChipTop
Version: I-2014.03
Date   : Wed Apr  9 04:30:04 2014
```

```
****************************************
Mode: default
Corner: default
Time unit: 1.00ns

Switching Activity Overview Statistics for 'ChipTop'
---------------------------------------------

Object Type    annotated  default  Total
---------------------------------------------

Nets              184     7460   7644
---------------------------------------------

Nets Driven by
---------------------------------------------

Primary Input       1      124    125
Memory              0        0      0
Black Box           0        0      0
Clock Gate          0       64     64
Tri-State           0        0      0
Sequential        183      859   1042
Combinational       0     6413   6413

---------------------------------------------

1
```

In the following example, the **report_switching_activity** command is used after the **read_saif** command. 3 objects now have type of activity as 'simulated'.

```
prompt> report_switching_activity
****************************************
Report : switching_activity
Design : ChipTop
Version: I-2014.03
Date   : Wed Apr  9 04:30:04 2014
****************************************
Mode: default
Corner: default
Time unit: 1.00ns

Switching Activity Overview Statistics for 'ChipTop'
---------------------------------------------------------------------------

Object Type    annotated  propagated  simulated  sta  default  Total
---------------------------------------------------------------------------

Nets              184         0         3    0     7457   7644
---------------------------------------------------------------------------

Nets Driven by
---------------------------------------------------------------------------

Primary Input       1         0         3    0      121    125
Memory              0         0         0    0        0      0
```

```
Black Box           0         0        0   0       0     0
Clock Gate          0         0        0   0      64    64
Tri-State           0         0        0   0       0     0
Sequential        183         0        0   0     859  1042
Combinational       0         0        0   0    6413  6413

---------------------------------------------------------------------------

1
```

In the following example the **report_switching_activity** command is used to get the switching activity information on the cells 'Multiplier' and 'InstDecode' and the subblocks of the design (using the *-hierarchy* option)

```
prompt> report_switching_activity \
  -cells {InstDecode Multiplier} -hierarchy
****************************************
Report : switching_activity
Design : ChipTop
Version: I-2014.03
Date   : Wed Apr  9 04:30:04 2014
****************************************
Mode: default
Corner: default
Time unit: 1.00ns

Switching Activity Overview Statistics for 'Multiplier'
---------------------------------------------------------------------------

Object Type    annotated  propagated  simulated  sta  default  Total
---------------------------------------------------------------------------

Nets               33         0        0   0     3951   3984
---------------------------------------------------------------------------

Nets Driven by
---------------------------------------------------------------------------

Primary Input       0         0        0   0        1      1
Memory              0         0        0   0        0      0
Black Box           0         0        0   0        0      0
Clock Gate          0         0        0   0        0      0
Tri-State           0         0        0   0        0      0
Sequential         33         0        0   0      160    193
Combinational       0         0        0   0     3790   3790

---------------------------------------------------------------------------

Multiplier         33         0        0   0     3951   3984
Multiplier/GENPP    0         0        0   0     1979   1979
---------------------------------------------------------------------------


Switching Activity Overview Statistics for 'InstDecode'
---------------------------------------------------------------------------

Object Type    annotated  propagated  simulated  sta  default  Total
---------------------------------------------------------------------------
```

```
Nets              36         0        0   0      46     82
--------------------------------------------------------------------------------

Nets Driven by
--------------------------------------------------------------------------------

Primary Input      1         0        0   0      32     33
Memory             0         0        0   0       0      0
Black Box          0         0        0   0       0      0
Clock Gate         0         0        0   0       0      0
Tri-State          0         0        0   0       0      0
Sequential        35         0        0   0       1     36
Combinational      0         0        0   0      13     13

--------------------------------------------------------------------------------

InstDecode        36         0        0   0      46     82
--------------------------------------------------------------------------------

1
```

In the following example the **report_switching_activity** command is used with the *-coverage* and *-hierarchy* options to get the switching activity coverage report for the cells 'InstDecode' and 'Multiplier' and subblocks of the design. The toggle rate limit has been set as 0.035 (using the *-toggle_rate_limit* option).

```
prompt> report_switching_activity -cells {InstDecode Multiplier} \
  -coverage -hierarchy -toggle_rate_limit 0.035
****************************************
Report : switching_activity
       -coverage
       -hierarchy
Design : ChipTop
Version: L-2016.03-SP5
Date   : Sun Aug 28 23:56:15 2016
****************************************
Mode: default
Corner: default
Time unit: 1ns
Fastest clock: clock with period 3 ns

Coverage is defined as the percent of nets with toggle rate greater than 0.035000
--------------------------------------------------------------------------------

Switching Activity Coverage for "InstDecode"
--------------------------------------------------------------------------------

Block                % Nets Covered     # Nets Covered       Total Nets

InstDecode                     1.22               1               82
--------------------------------------------------------------------------------

Switching Activity Coverage for "Multiplier"
--------------------------------------------------------------------------------

Block                % Nets Covered     # Nets Covered       Total Nets
```

```
Multiplier                    0.03            1          3984
Multiplier/GENPP              0.00            0          1979
---------------------------------------------------------------------------
1
```

In the following example the **report_switching_activity** command is used with the *-list_low_activity* option to get the list of low activity nets for the cell 'Multiplier'. The toggle rate limit has been set as 0.001 (using the *-toggle_rate_limit* option).

```
prompt> report_switching_activity -list_low_activity \
  -cells {Multiplier} -toggle_rate_limit 0.001
****************************************
Report : switching_activity
       -list_low_activity
Design : ChipTop
Version: L-2016.03-SP5
Date   : Mon Aug 29 00:03:34 2016
****************************************
Mode: default
Corner: default
Time unit: 1ns
Fastest clock: clock with period 3 ns

Low activity nets are the nets having toggle rate less than or equal to 0.001000
---------------------------------------------------------------------------

Switching Activity Overview Statistics for 'Multiplier'
---------------------------------------------------------------------------

Object Type    simulated  annotated  sta  implied  propagated  estimated  default  Total
-------------------------------------------------------------------------------------

Nets               0        5    1       0         0         0     3978   3984
-------------------------------------------------------------------------------------

Nets Driven by
-------------------------------------------------------------------------------------

Primary Input        0         0    1       0         0         0       0      1
Memory               0         0    0       0         0         0       0      0
Black Box            0         0    0       0         0         0       0      0
Clock Gate           0         0    0       0         0         0       0      0
Tri-State            0         0    0       0         0         0       0      0
Sequential           0         0    0       0         0         0     193    193
Combinational        0         5    0       0         0         0    3785   3790

-------------------------------------------------------------------------------------

List of low activity nets:
Multiplier/*Logic0*
Multiplier/VSS
Multiplier/VDD90_SD
Multiplier/GENPP/VSS
Multiplier/GENPP/VDD90_SD
-----------------------------------------
5 net(s) with low activity

1
```

In the following example, the **report_switching_activity** command is used with the *-essential* option to report the activity for the essential activity points in the design.

```
prompt> report_switching_activity -essential
Essential activity is complete
****************************************
Report : switching_activity
       -essential
Design : ChipTop
Version: L-2016.03-SP5-BETA
Date   : Mon Aug 29 00:03:34 2016
****************************************
Mode: default
Corner: default
Time unit: 1ns
Fastest clock: clock with period 3 ns

Essential Switching Activity Overview Statistics for 'ChipTop'
---------------------------------------------------------------------------------------------

Essential Group  simulated  annotated  sta  implied  propagated  estimated  default  Total
---------------------------------------------------------------------------------------------

Nets                0       80    4      91       7071        0      398    7644
---------------------------------------------------------------------------------------------

Essential Group
---------------------------------------------------------------------------------------------

Primary Input        0        0    4       0         0         0      121     125
Black Box            0        0    0       0         0         0      0       0
Edge Sensitive       0        0    0       0      1042         0      0    1042
Level Sensitive      0        0    0       0         0         0      0       0
Logical Cycle        0        0    0       0         0         0      0       0
<combo>              0        0    0       0         0         0      0       0
non essential        0       80    0      91      6029         0    277    6477

---------------------------------------------------------------------------------------------

1
```

SEE ALSO

get_switching_activity(2)
read_saif(2)
reset_switching_activity(2)
set_switching_activity(2)

report_synlib

Displays information about synthetic libraries.

SYNTAX

status **report_synlib**
[**-modules** *module_list*]
[**-nosplit**]
library

Data Types

library string
module_list list

ARGUMENTS

library

Specifies the name of the library to be reported. This library must already be read into the tool except the *standard* and *dw_foundation*.

-modules ***module_list***

Specifies a list of modules that are reported. The default is to report information about all modules in the synthetic library. Wildcard pattern can be used in each element of the list.

-nosplit

Does not split lines if column overflows.

DESCRIPTION

The **report_synlib** command displays information about synthetic libraries.

The synthetic library report displays the following information in the following order:

- A list of the operators and their pins
- A list of the modules with their pins, parameters, attributes, implementations and bindings

By default, all the modules are reported. If you specify the **-modules** option, the **report_synlib** command reports only the listed

modules and the associated implementations and bindings.

EXAMPLES

The following report displays the *DW01_add* and *DW01_addsub* modules and their implementations in the *standard* library.

```
prompt> report_synlib -modules {DW01_fp_add*} standard
****************************************
Report : report_synlib
Version: 2.5.9
Date   : Sat May 13 23:45:53 2017
****************************************

Library Name    : dw_foundation
Tool Created    : K-2015.06
Date Created    : 05.13.17
Library Version : D-2010.03:L-2016.03-DWBB_201603.0:88108c49

Module Parameters

                Design   HDL
  Module Name     Library  Parameters          Parameters
 -------------------------------------------------------------------------------
  DW_fp_addsub    DW02     sig_width = 23      fp_size = exp_width+sig_width+1
                           exp_width = 8
                           ieee_compliance = 0
  DW_fp_add       DW02     sig_width = 23      fp_size = exp_width+sig_width+1
                           exp_width = 8
                           ieee_compliance = 0
  DW_fp_addsub_DG DW02     sig_width = 23      fp_size = exp_width+sig_width+1
                           exp_width = 8
                           ieee_compliance = 0
  DW_fp_add_DG    DW02     sig_width = 23      fp_size = exp_width+sig_width+1
                           exp_width = 8
                           ieee_compliance = 0

Module Pins

                                  Default  Stall  Pin
  Module Name     Pins    Direction  Width    Value    Pin    Attribute
 -------------------------------------------------------------------------------
  DW_fp_addsub    a       input     fp_size
                  b       input     fp_size
                  rnd     input     3
                  op      input     1
                  z       output    fp_size
                  status  output    8
  DW_fp_add       a       input     fp_size
                  b       input     fp_size
                  rnd     input     3
                  z       output    fp_size
                  status  output    8
  DW_fp_addsub_DG a       input     fp_size
```

```
            b       input     fp_size
            rnd     input     3
            op      input     1
            DG_ctrl input     1
            z       output    fp_size
            status  output    8
  DW_fp_add_DG  a       input     fp_size
            b       input     fp_size
            rnd     input     3
            DG_ctrl input     1
            z       output    fp_size
            status  output    8

Module Implementations

  Module Name   Implemenations  Attributes          Parameters
 ---------------------------------------------------------------------------------------------------------
  DW_fp_addsub  rtl             license = DesignWare  legal = (sig_width >= 2) && (sig_width <= 253) && (exp_width >= 3) && (exp_wi

              str             license = DesignWare  legal = (sig_width >= 2) && (sig_width <= 253) && (exp_width >= 3) && (exp_width <= 3

  DW_fp_add     rtl           license = DesignWare  legal = (sig_width >= 2) && (sig_width <= 253) && (exp_width >= 3) && (exp_widtl


1
```

SEE ALSO

report_lib(2)

report_taps

Reports information on tap points.

SYNTAX

string **report_taps**
[-static_current]
[*taps*]

Data Types

list *taps*

ARGUMENTS

-static_current

Report static current for each tap point. The info. is only available from static rail analysis.

taps

Specifies a list of taps to be reported. Items within the list can be name patterns or collections of taps.

DESCRIPTION

The **report_taps** command displays information about the tap points in the current session context. By default, the command generates a report that includes all current tap points in the session context. Options can be used to limit the report to a subset of the taps. The report contains the following information about each tap:

- Name
- Type (auto_pg, top_pg, cell_pg, absolute_coord, cell_coord, libcell_pg and libcell_coord)
- Associated net
- Absolute {X Y} coordinate
- Layer number from the technology file (.tf)
- Associated parasitics
- Validity of the tap point location

- Static current

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example shows how to report all taps:

```
prompt> report_taps

Name   Type    Net    Point        Layer  Is_valid
------------------------------------------------------------------------
Tap_0  cell_coord    VDD    17.600 9.000  31    true
Tap_1  cell_coord    VDD    9.225  21.603 32    true
```

The following example shows how to report partial taps:

```
prompt> report_taps [get_taps Tap_0]

Name   Type    Net    Point        Layer  Is_valid
------------------------------------------------------------------------
Tap_0  cell_coord    VDD    17.600 9.000  31    true

prompt> report_taps Tap_1

Name   Type    Net    Point        Layer  Is_valid
------------------------------------------------------------------------
Tap_1  cell_coord    VDD    9.225  21.603 32    true
```

The following example shows associated parasitics will be shown if any tap has valid RLC:

```
prompt> report_taps

Name  Type  Net Point   Layer R   L   C   Is_valid
-------------------------------------------------------------------------------------------
Tap_5  absolute_coord  VDD 18.225  5.270 32  R(1.000)  L(2.000)  C(3.000)  true
Tap_7  absolute_coord  VSS 22.265  4.475 31  R(3.000)  L(2.000)  C(1.000)  true
```

The following example shows how to report tap static current:

```
prompt> report_taps -static_current

Name  Type  Net Point   Layer Static_Current
-----------------------------------------------------------------------
Tap_28    VSS 521.100 409.740 31  -0.0013228
Tap_29    VSS 521.100 0.200 31  -0.00153719
Tap_30    VSS 420.300 409.740 31  -0.000945483
Tap_31    VSS 420.300 0.200 31  -0.00100271
Tap_32    VSS 319.500 409.740 31  -0.000893119
Tap_33    VSS 319.500 0.200 31  -0.00102184
Tap_34    VSS 218.700 409.740 31  -0.00105649
Tap_35    VSS 218.700 0.200 31  -0.00112674
```

```
Tap_36   VSS 117.900 409.740 31  -0.0012568
Tap_37   VSS 117.900 0.200 31  -0.00133558
Tap_38   VDD 672.300 409.740 31  0
Tap_39   VDD 672.300 0.200 31  0
Tap_40   VDD 571.500 409.740 31  0.00109758
Tap_41   VDD 571.500 0.200 31  0.00126164
Tap_42   VDD 470.700 409.740 31  0.000824849
Tap_43   VDD 470.700 0.200 31  0.000894117
Tap_44   VDD 369.900 409.740 31  0.000800109
```

SEE ALSO

get_taps(2)
create_taps(2)
remove_taps(2)

report_target_library_subset

Reports target library subsets on the top block and hierarchical cells.

SYNTAX

status **report_target_library_subset**
[-objects *objects*]
[-top]
[-all]
[-nosplit]

Data Types

objects list or collection

ARGUMENTS

-objects *objects*

Specifies the hierarchical cells for which to report the target library subsets. The cells must be in the current block.

If you do not specify this option, the **-top** option, or the **-all** option, the tool uses the **-top** option by default. Specifying the **-top** and **-objects** options is also supported.

-top

Reports the target library subset on the top block.

If you do not specify this option, the **-objects** option, or the **-all** option, the tool uses the **-top** option by default. Specifying the **-top** and **-objects** options is also supported.

-all

Reports the target library subset on each hierarchy which has an explicitly assigned target library subset. The **-all** option cannot be used with the **-top** or **-objects** options.

-nosplit

Does not split lines if column overflows.

DESCRIPTION

This command reports the target library subsets for the top block and hierarchical cells. This command only reports the subsets which were explicitly set on the objects with the set_target_library_subset command. This command does not report subsets which are inherited from an ancestor hierarchical cell.

EXAMPLES

The following example reports the target library subset for the top hierarchy.

```
prompt> report_target_library_subset -top
```

The following example reports the target library subset for a hierarchical cell.

```
prompt> report_target_library_subset -objects [get_cells top/mid]
```

SEE ALSO

set_target_library_subset(2)
remove_target_library_subset(2)

report_tech_diff

Prints a report of difference between two Techs.

SYNTAX

status **report_tech_diff**
-tech *cmp_tech*
-tech_file *cmp_tech_file*
-ref_tech *ref_tech*
-ref_tech_file *ref_tech_file*

Data Types

tech	collection
tech_file	string
ref_tech	collection
ref_tech_file	string

ARGUMENTS

-tech *tech*

The tech object which needs to be compared against reference.This option and the *-tech_file* option are mutually exclusive.

-tech_file *tech_file*

The tech file which needs to be compared against reference.This option and the *-tech* option are mutually exclusive.

[-tech | -tech_file] is a compulsory option.

-ref_tech *ref_tech*

For reading in tech object who is reference for the comparison of new tech.This option and the *-ref_tech_file* option are mutually exclusive.

-ref_tech_file {*ref_tech_file*}

For reading in tech file who is reference for the comparison of new tech.This option and the *-ref_tech* option are mutually exclusive.

[[-ref_tech] | [-ref_file_tech]] is an optional option. If -ref_tech or -ref_file_tech is not provided, command will take tech of current lib, if current lib is not present it will give Error that no reference tech is present for comparison.

DESCRIPTION

The **report_tech_diff** command will print a detailed report of the differences between the new tech and the reference tech.

The **report_tech_diff** command will compare two techs, regardless of whether they are ASCII (tech file) or Tech database(object):

1. Two ASCII tech files.

2. Two tech databases.

3. An ASCII tech file and a tech database.

The **report_tech_diff** command will produce a detailed report of the differences between the new tech and the reference tech containing the list of all added tech objects, all removed tech objects, and all changed objects of the given type, with the changes detailed for changed tech objects.

EXAMPLES

The following example uses the **-tech_file** and **-ref_tech_file** options for equivalent comparison of techs.

```
prompt> report_tech_diff -tech_file tech_file.tf -ref_tech_file ref_tech_file.tf

************************************************
Report : Technology Files Differences
Version:
Date   :
************************************************

List of Technology Section Differences

Property Name       Change Type    Reference Tech   Tech
-------------------------------------------------------------------------
unitTimeName        Changed       ns             ms
timePrecision       Removed       1000           -
currentPrecision    Added         -              1000
 ...

List of Layer Section Differences

Property Name    Change Type       Reference Tech    Tech
-------------------------------------------------------------------------
Comparing NW
visible          Changed          1              0
Pattern          Changed          blank          enter
lineStyle        Added            -              dash
color            Removed          21             -
 ...
Comparing OA                                    no difference
Comparing OD
color            Changed          19             21
```

```
 ...

Added Layer Name
---------------------
PP
NP
 ...

Removed Layer Name
---------------------
MO
PO
 ...

List of Color Section Differences
 ...

Techs are different.
```

SEE ALSO

get_techs(2)

report_test_assume

Displays the value set on the pins by the **set_test_assume** command.

SYNTAX

status **report_test_assume**

ARGUMENTS

The **report_test_assume** command has no arguments.

DESCRIPTION

This command displays the pins and the value they are set to by the **set_test_assume** command on the current design. It reports the test assume value of either 1 or 0.

EXAMPLES

The following is an example of the **report_test_assume** command:

```
prompt> report_test_assume
****************************************
Report : Test_Assume Specification
Design : top
Version: P-2019.03
Date   : Thu Feb  7 07:59:36 2019
****************************************

PIN NAME                    Test Assume Value
--------                  ------------------
ff1/CD                      0
ff6/Q                     1
ff2/CD                      1
```

SEE ALSO

set_test_assume(2)

report_test_point_configuration

Displays options specified by the **set_testability_configuration** command.

SYNTAX

status **report_testability_configuration**

ARGUMENTS

The **report_testability_configuration** command has no arguments.

DESCRIPTION

This command displays the options specified by the **set_testability_configuration** command on the current design.

EXAMPLES

The following example shows a testability report.

```
prompt> report_testability_configuration
****************************************
Report : Test Points Specification
Design : top
Version: P-2019.03
Date   : Thu Feb  7 08:00:40 2019
****************************************

----------------------------------------
TEST MODE: all_dft
VIEW     : Specification
----------------------------------------
Test Point Analysis Target::        random_resistant
Control Signal:
Clock Signal:

Test Point Analysis Target::        untestable_logic
```

Control Signal:
Clock Signal:

SEE ALSO

set_testability_configuration(2)

report_threshold_voltage_group

Reports statistics on cell count and area by threshold voltage group names.

SYNTAX

int **report_threshold_voltage_groups**
[-verbose]
[-significant_digits *digits*]
[-nosplit]
[-summary]
[-detailed *cell_type*]
[-datapath_cells_only]
[-standard_cells_only]
[-hierarchy]
[-hier_vt_groups *group_list*]
[-hier_threshold *digits*]
[*cell_list*]

Data Types

digits int
cell_type list
cell_list list
group_list list

ARGUMENTS

-verbose

Lists all of the cells belonging to each threshold voltage group. If this option is omitted, the default summary report shows only the number and percentage of cells.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 through 13. The default is 2. Using this option overrides the value set by the **report_default_significant_digits** variable.

-nosplit

Prevents line splitting and facilitates writing scripts to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-summary

Prints the old report format. This format reports the number, area, and percentage of cells in each threshold voltage group for black-box and non-black-box cells in each group.

-detailed *cell_type*

Specifies the cell types for which detailed report is to be generated. Acceptable types are repeater, register and clock_network.

-datapath_cells_only

Reports only for cells in the data path excluding clock_network, macros and blackboxes

-standard_cells_only

Reports only for cells in data path and clock network

-hierarchy

Prints the hierarchical report for all vtGroups present in the design

-hier_vt_groups *group_list*

Prints the hierarchical report for user specified vtGroups in the design.

-hier_threshold *digits*

Limits the report to hierarchical cells that has leaf cells more than the specified threshold. The default value is 10000.

cell_list

Specifies a list of cells on which to report the threshold voltage groups. If not specified, all cells in the **current_design** are considered.

DESCRIPTION

This command reports the number, area, and percentage of cells in each threshold voltage group for the specified list of cells or designs. The report also shows separately, the number, area, and percentage of repeater, combinational, register, sequential, clock_network, macro and physical_only cells in each group.

The threshold voltage group is defined in the technology libraries with the **threshold_voltage_group** attribute on the cell, or with the **default_threshold_voltage_group** attribute on the library. You can also set the attributes on the objects using the **set_attribute** command. The attributes can be any string values, but the values must be related to the threshold voltage of the cell to be meaningful. Any cells in the design that have no threshold voltage group attribute are included in the "undefined" group. Cells in the abstract block will not be reported.

The threshold voltage or vth groups that are to be considered as low Vth are specified using the **set_threshold_voltage_group_type** command.

If specified, these low Vth groups are marked by a letter "L" in the last column of their corresponding rows in the report. The total number, area, and percentage of cells in all of these low vth groups is also reported.

If the command set_max_lvth_percentage is used to set the max lvt percentage, the value will be displayed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This command generates a summary of cells in each threshold voltage group:

```
prompt> set_max_lvth_percentage 10
prompt> report_threshold_voltage_groups -summary
********************************************************************************
                   Threshold Voltage Group Report

Threshold           All            Blackbox         Non-blackbox
Voltage Group        cells            cells             cells
********************************************************************************
hvt               34   (7.87%)        1  (50.00%)        33   (7.67%)
lvt              342  (79.17%)        1  (50.00%)       341  (79.30%) L
svt               56  (12.96%)        0   (0.00%)        56  (13.02%) L
********************************************************************************
Total            432 (100.00%)        2 (100.00%)       430 (100.00%)
Low vth groups      398  (92.13%)        1  (50.00%)       397  (92.33%)

Threshold           All            Blackbox         Non-blackbox
Voltage Group       cell area          cell area          cell area
********************************************************************************
hvt            185.36   (7.82%)     20.00  (66.67%)     165.36   (7.06%)
lvt           1975.60  (83.30%)     10.00  (33.33%)    1965.60  (83.94%) L
svt            210.60   (8.88%)      0.00   (0.00%)     210.60   (8.99%) L
********************************************************************************
Total          2371.56 (100.00%)     30.00 (100.00%)    2341.56 (100.00%)
Low vth groups   2186.20  (92.18%)     10.00  (33.33%)    2176.20  (92.94%)

Note: L denotes cells of low vth groups
%LVT max          : 10.00
1

prompt> report_threshold_voltage_groups

Cell Count Report
----------------------------------------------------------------------------------------------------------------------------------------------------------


Vth Group Names        All (%)     Repeaters (%)     Combinational (%)     Registers (%)     Sequentials (%)       Clock_network (%)
----------------------------------------------------------------------------------------------------------------------------------------------------------
hvt             23  (85.19%)       1   (3.70%)       18  (66.67%)       0   (0.00%)       4  (14.81%)       0   (0.00%)       0   (0.00%)       0
lvt             4  (14.81%)       4  (14.81%)       0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0.00%)       0   ((
----------------------------------------------------------------------------------------------------------------------------------------------------------
Total            27 (100.00%)       5  (18.52%)       18  (66.67%)       0   (0.00%)       4  (14.81%)       0   (0.00%)       0   (0.00%)
Low vth groups       4  (14.81%)       4  (14.81%)       0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0.00%)
----------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------

Cell Area Report
----------------------------------------------------------------------------------------------------------------------------------------------------------


Vth Group Names        All (%)     Repeaters (%)     Combinational (%)     Registers (%)     Sequentials (%)       Clock_network (%)
```

```
--------------------------------------------------------------------------------------------------------------------------------------------------------------
hvt              434.53  (95.52%)     3.39   (0.75%)   249.52  (54.85%)     0.00   (0.00%)   181.62  (39.93%)     0.00   (0.00%)     0.00  (
lvt               20.37   (4.48%)    20.37   (4.48%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00  (0.00%
--------------------------------------------------------------------------------------------------------------------------------------------------------------
Total            454.90 (100.00%)    23.76   (5.22%)   249.52  (54.85%)     0.00   (0.00%)   181.62  (39.93%)     0.00   (0.00%)     0.00
Low vth groups    20.37   (4.48%)    20.37   (4.48%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00
--------------------------------------------------------------------------------------------------------------------------------------------------------------

Legend
L:  Low Vth group specified by user
1
```

prompt> **report_threshold_voltage_groups -detailed repeater**

```
Cell Count Report
--------------------------------------------------------------------------------------------------------------------------------------------------------------


Vth Group Names         All (%)     Repeaters (%)     Combinational (%)     Registers (%)     Sequentials (%)       Clock_network (%)
--------------------------------------------------------------------------------------------------------------------------------------------------------------
hvt               23  (85.19%)       1   (3.70%)       18  (66.67%)       0   (0.00%)       4  (14.81%)       0   (0.00%)       0   (0.00%)       0
lvt                4  (14.81%)       4  (14.81%)        0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (
--------------------------------------------------------------------------------------------------------------------------------------------------------------
Total             27 (100.00%)       5  (18.52%)       18  (66.67%)       0   (0.00%)       4  (14.81%)       0   (0.00%)       0   (0.00%)
Low vth groups     4  (14.81%)       4  (14.81%)        0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0.00%)
--------------------------------------------------------------------------------------------------------------------------------------------------------------

Repeater detailed Cell Count Report
--------------------------------------------------------------------------------------------


Vth Group Names       All (%)       Buffer (%)        Inverter (%)   Hold buffer (%)
------------------------------------------------------------------------------------------
hvt                1  (20.00%)       0   (0.00%)       1  (20.00%)       0   (0.00%)
lvt                4  (80.00%)       0   (0.00%)       4  (80.00%)       0   (0.00%)  L
------------------------------------------------------------------------------------------
Total              5 (100.00%)       0   (0.00%)       5 (100.00%)       0   (0.00%)
Low vth groups     4  (80.00%)       0   (0.00%)       4  (80.00%)       0   (0.00%)
------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------

Cell Area Report
--------------------------------------------------------------------------------------------------------------------------------------------------------------


Vth Group Names         All (%)     Repeaters (%)     Combinational (%)     Registers (%)     Sequentials (%)       Clock_network (%)
--------------------------------------------------------------------------------------------------------------------------------------------------------------
hvt              434.53  (95.52%)     3.39   (0.75%)   249.52  (54.85%)     0.00   (0.00%)   181.62  (39.93%)     0.00   (0.00%)     0.00  (
lvt               20.37   (4.48%)    20.37   (4.48%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00  (0.00%
--------------------------------------------------------------------------------------------------------------------------------------------------------------
Total            454.90 (100.00%)    23.76   (5.22%)   249.52  (54.85%)     0.00   (0.00%)   181.62  (39.93%)     0.00   (0.00%)     0.00
Low vth groups    20.37   (4.48%)    20.37   (4.48%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00
--------------------------------------------------------------------------------------------------------------------------------------------------------------

Repeater detailed Cell Area Report
--------------------------------------------------------------------------------------------
```

```
Vth Group Names       All (%)       Buffer (%)        Inverter (%)   Hold buffer (%)
-------------------------------------------------------------------------------------------
hvt              3.39  (14.29%)     0.00   (0.00%)     3.39  (14.29%)     0.00   (0.00%)
lvt             20.37  (85.71%)     0.00   (0.00%)    20.37  (85.71%)     0.00   (0.00%)  L
-------------------------------------------------------------------------------------------
Total           23.76 (100.00%)     0.00   (0.00%)    23.76 (100.00%)     0.00   (0.00%)
Low vth groups  20.37  (85.71%)     0.00   (0.00%)    20.37  (85.71%)     0.00   (0.00%)
-------------------------------------------------------------------------------------------

Legend
L:  Low Vth group specified by user
1
```

prompt> **report_threshold_voltage_groups -hierarchy -hier_threshold 30**

```
Hierarchy                |       Total Cell Detail       |           saed32cell_hvt            |              undefined              |
                         |      Cell#       Area Ratio% |      Cell#       Area Block% Design% |      Cell#       Area Block% Design% |
-------------------------------------------------------------------------------------------------------------------------------------------
des_unit                 |       6647      390.37 100.00 |       194      390.37 100.00  100.00 |       6453       0.00   0.00    0.00 |
 ctl                     |        322       0.00   0.00 |         0       0.00   0.00    0.00 |        322       0.00   0.00    0.00 |
 de_a                    |        115      59.47  15.23 |        30      59.47 100.00   15.23 |         85       0.00   0.00    0.00 |
  add_85                 |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
   U0                    |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
 de_b                    |         75       0.00   0.00 |         0       0.00   0.00    0.00 |         75       0.00   0.00    0.00 |
 de_encrypt              |       2769      76.24  19.53 |        37      76.24 100.00   19.53 |       2732       0.00   0.00    0.00 |
  add_90                 |         37      76.24  19.53 |        37      76.24 100.00   19.53 |          0       0.00   0.00    0.00 |
   U0                    |         37      76.24  19.53 |        37      76.24 100.00   19.53 |          0       0.00   0.00    0.00 |
 de_c                    |         75       0.00   0.00 |         0       0.00   0.00    0.00 |         75       0.00   0.00    0.00 |
 de_d                    |        126      59.47  15.23 |        30      59.47 100.00   15.23 |         96       0.00   0.00    0.00 |
  add_90                 |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
   U0                    |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
 dd_a                    |        115      59.47  15.23 |        30      59.47 100.00   15.23 |         85       0.00   0.00    0.00 |
  add_85                 |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
   U0                    |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
 dd_b                    |         75       0.00   0.00 |         0       0.00   0.00    0.00 |         75       0.00   0.00    0.00 |
 dd_decrypt              |       2774      76.24  19.53 |        37      76.24 100.00   19.53 |       2737       0.00   0.00    0.00 |
  add_90                 |         37      76.24  19.53 |        37      76.24 100.00   19.53 |          0       0.00   0.00    0.00 |
   U0                    |         37      76.24  19.53 |        37      76.24 100.00   19.53 |          0       0.00   0.00    0.00 |
 dd_c                    |         75       0.00   0.00 |         0       0.00   0.00    0.00 |         75       0.00   0.00    0.00 |
 dd_d                    |        126      59.47  15.23 |        30      59.47 100.00   15.23 |         96       0.00   0.00    0.00 |
  add_90                 |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
   U0                    |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
1
```

SEE ALSO

set_attribute(2)
set_max_lvth_percentage(2)
set_threshold_voltage_group_type(2)

report_threshold_voltage_groups

Reports statistics on cell count and area by threshold voltage group names.

SYNTAX

int **report_threshold_voltage_groups**
[-verbose]
[-significant_digits *digits*]
[-nosplit]
[-summary]
[-detailed *cell_type*]
[-datapath_cells_only]
[-standard_cells_only]
[-hierarchy]
[-hier_vt_groups *group_list*]
[-hier_threshold *digits*]
[*cell_list*]

Data Types

digits int
cell_type list
cell_list list
group_list list

ARGUMENTS

-verbose

Lists all of the cells belonging to each threshold voltage group. If this option is omitted, the default summary report shows only the number and percentage of cells.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are from 0 through 13. The default is 2. Using this option overrides the value set by the **report_default_significant_digits** variable.

-nosplit

Prevents line splitting and facilitates writing scripts to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-summary

Prints the old report format. This format reports the number, area, and percentage of cells in each threshold voltage group for black-box and non-black-box cells in each group.

-detailed *cell_type*

Specifies the cell types for which detailed report is to be generated. Acceptable types are repeater, register and clock_network.

-datapath_cells_only

Reports only for cells in the data path excluding clock_network, macros and blackboxes

-standard_cells_only

Reports only for cells in data path and clock network

-hierarchy

Prints the hierarchical report for all vtGroups present in the design

-hier_vt_groups *group_list*

Prints the hierarchical report for user specified vtGroups in the design.

-hier_threshold *digits*

Limits the report to hierarchical cells that has leaf cells more than the specified threshold. The default value is 10000.

cell_list

Specifies a list of cells on which to report the threshold voltage groups. If not specified, all cells in the **current_design** are considered.

DESCRIPTION

This command reports the number, area, and percentage of cells in each threshold voltage group for the specified list of cells or designs. The report also shows separately, the number, area, and percentage of repeater, combinational, register, sequential, clock_network, macro and physical_only cells in each group.

The threshold voltage group is defined in the technology libraries with the **threshold_voltage_group** attribute on the cell, or with the **default_threshold_voltage_group** attribute on the library. You can also set the attributes on the objects using the **set_attribute** command. The attributes can be any string values, but the values must be related to the threshold voltage of the cell to be meaningful. Any cells in the design that have no threshold voltage group attribute are included in the "undefined" group. Cells in the abstract block will not be reported.

The threshold voltage or vth groups that are to be considered as low Vth are specified using the **set_threshold_voltage_group_type** command.

If specified, these low Vth groups are marked by a letter "L" in the last column of their corresponding rows in the report. The total number, area, and percentage of cells in all of these low vth groups is also reported.

If the command set_max_lvth_percentage is used to set the max lvt percentage, the value will be displayed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This command generates a summary of cells in each threshold voltage group:

```
prompt> set_max_lvth_percentage 10
prompt> report_threshold_voltage_groups -summary
****************************************************************************
                Threshold Voltage Group Report

Threshold            All            Blackbox        Non-blackbox
Voltage Group        cells          cells           cells
****************************************************************************
hvt              34  (7.87%)       1  (50.00%)      33  (7.67%)
lvt             342  (79.17%)      1  (50.00%)     341  (79.30%) L
svt              56  (12.96%)      0  (0.00%)      56  (13.02%) L
****************************************************************************
Total           432 (100.00%)      2 (100.00%)     430 (100.00%)
Low vth groups  398  (92.13%)      1  (50.00%)     397  (92.33%)

Threshold            All            Blackbox        Non-blackbox
Voltage Group        cell area      cell area       cell area
****************************************************************************
hvt           185.36  (7.82%)    20.00  (66.67%)   165.36  (7.06%)
lvt          1975.60  (83.30%)   10.00  (33.33%)  1965.60  (83.94%) L
svt           210.60  (8.88%)    0.00  (0.00%)    210.60  (8.99%) L
****************************************************************************
Total        2371.56 (100.00%)   30.00 (100.00%)  2341.56 (100.00%)
Low vth groups  2186.20  (92.18%)   10.00  (33.33%)  2176.20  (92.94%)

Note: L denotes cells of low vth groups
%LVT max          : 10.00
1

prompt> report_threshold_voltage_groups

Cell Count Report
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------


Vth Group Names        All (%)    Repeaters (%)    Combinational (%)    Registers (%)    Sequentials (%)      Clock_network (%)
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
hvt             23  (85.19%)      1  (3.70%)      18  (66.67%)      0  (0.00%)      4  (14.81%)      0  (0.00%)      0  (0.00%)      0
lvt             4  (14.81%)      4  (14.81%)      0  (0.00%)      0  (0.00%)      0  (0.00%)      0  (0.00%)      0  (0.00%)      0  (
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Total           27 (100.00%)      5  (18.52%)      18  (66.67%)      0  (0.00%)      4  (14.81%)      0  (0.00%)      0  (0.00%)
Low vth groups      4  (14.81%)      4  (14.81%)      0  (0.00%)      0  (0.00%)      0  (0.00%)      0  (0.00%)      0  (0.00%)
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Cell Area Report
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------


Vth Group Names        All (%)    Repeaters (%)    Combinational (%)    Registers (%)    Sequentials (%)      Clock_network (%)
```

```
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
hvt              434.53  (95.52%)     3.39   (0.75%)   249.52  (54.85%)     0.00   (0.00%)   181.62  (39.93%)     0.00   (0.00%)     0.00  (
lvt               20.37   (4.48%)    20.37   (4.48%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00  (0.00%
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
Total            454.90 (100.00%)    23.76   (5.22%)   249.52  (54.85%)     0.00   (0.00%)   181.62  (39.93%)     0.00   (0.00%)     0.00
Low vth groups    20.37   (4.48%)    20.37   (4.48%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00
-------------------------------------------------------------------------------------------------------------------------------------------------------------------

Legend
L:  Low Vth group specified by user
1

prompt> report_threshold_voltage_groups -detailed repeater

Cell Count Report
-------------------------------------------------------------------------------------------------------------------------------------------------------------------


Vth Group Names         All (%)     Repeaters (%)     Combinational (%)     Registers (%)     Sequentials (%)       Clock_network (%)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
hvt               23  (85.19%)       1   (3.70%)       18  (66.67%)       0   (0.00%)       4  (14.81%)       0   (0.00%)       0   (0.00%)       0
lvt                4  (14.81%)       4  (14.81%)        0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
Total             27 (100.00%)       5  (18.52%)       18  (66.67%)       0   (0.00%)       4  (14.81%)       0   (0.00%)       0   (0.00%)
Low vth groups     4  (14.81%)       4  (14.81%)        0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0.00%)       0   (0.00%)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------

Repeater detailed Cell Count Report
-----------------------------------------------------------------------------------------------


Vth Group Names       All (%)       Buffer (%)        Inverter (%)   Hold buffer (%)
-----------------------------------------------------------------------------------------------
hvt                1  (20.00%)       0   (0.00%)       1  (20.00%)       0   (0.00%)
lvt                4  (80.00%)       0   (0.00%)       4  (80.00%)       0   (0.00%)  L
-----------------------------------------------------------------------------------------------
Total              5 (100.00%)       0   (0.00%)       5 (100.00%)       0   (0.00%)
Low vth groups     4  (80.00%)       0   (0.00%)       4  (80.00%)       0   (0.00%)
-----------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------

Cell Area Report
-------------------------------------------------------------------------------------------------------------------------------------------------------------------


Vth Group Names         All (%)     Repeaters (%)     Combinational (%)     Registers (%)     Sequentials (%)       Clock_network (%)
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
hvt              434.53  (95.52%)     3.39   (0.75%)   249.52  (54.85%)     0.00   (0.00%)   181.62  (39.93%)     0.00   (0.00%)     0.00  (
lvt               20.37   (4.48%)    20.37   (4.48%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00  (0.00%
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
Total            454.90 (100.00%)    23.76   (5.22%)   249.52  (54.85%)     0.00   (0.00%)   181.62  (39.93%)     0.00   (0.00%)     0.00
Low vth groups    20.37   (4.48%)    20.37   (4.48%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00   (0.00%)     0.00
-------------------------------------------------------------------------------------------------------------------------------------------------------------------

Repeater detailed Cell Area Report
-----------------------------------------------------------------------------------------------
```

```
Vth Group Names       All (%)       Buffer (%)        Inverter (%)   Hold buffer (%)
-------------------------------------------------------------------------------------------
hvt              3.39  (14.29%)     0.00   (0.00%)     3.39  (14.29%)     0.00   (0.00%)
lvt             20.37  (85.71%)     0.00   (0.00%)    20.37  (85.71%)     0.00   (0.00%)  L
-------------------------------------------------------------------------------------------
Total           23.76 (100.00%)     0.00   (0.00%)    23.76 (100.00%)     0.00   (0.00%)
Low vth groups  20.37  (85.71%)     0.00   (0.00%)    20.37  (85.71%)     0.00   (0.00%)
-------------------------------------------------------------------------------------------

Legend
L:  Low Vth group specified by user
1
```

prompt> **report_threshold_voltage_groups -hierarchy -hier_threshold 30**

```
Hierarchy                |       Total Cell Detail       |           saed32cell_hvt            |              undefined              |
                         |      Cell#       Area Ratio% |      Cell#       Area Block% Design% |      Cell#       Area Block% Design% |
-------------------------------------------------------------------------------------------------------------------------------------------
des_unit                 |       6647      390.37 100.00 |       194      390.37 100.00  100.00 |       6453       0.00   0.00    0.00 |
 ctl                     |        322       0.00   0.00 |         0       0.00   0.00    0.00 |        322       0.00   0.00    0.00 |
 de_a                    |        115      59.47  15.23 |        30      59.47 100.00   15.23 |         85       0.00   0.00    0.00 |
  add_85                 |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
   U0                    |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
 de_b                    |         75       0.00   0.00 |         0       0.00   0.00    0.00 |         75       0.00   0.00    0.00 |
 de_encrypt              |       2769      76.24  19.53 |        37      76.24 100.00   19.53 |       2732       0.00   0.00    0.00 |
  add_90                 |         37      76.24  19.53 |        37      76.24 100.00   19.53 |          0       0.00   0.00    0.00 |
   U0                    |         37      76.24  19.53 |        37      76.24 100.00   19.53 |          0       0.00   0.00    0.00 |
 de_c                    |         75       0.00   0.00 |         0       0.00   0.00    0.00 |         75       0.00   0.00    0.00 |
 de_d                    |        126      59.47  15.23 |        30      59.47 100.00   15.23 |         96       0.00   0.00    0.00 |
  add_90                 |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
   U0                    |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
 dd_a                    |        115      59.47  15.23 |        30      59.47 100.00   15.23 |         85       0.00   0.00    0.00 |
  add_85                 |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
   U0                    |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
 dd_b                    |         75       0.00   0.00 |         0       0.00   0.00    0.00 |         75       0.00   0.00    0.00 |
 dd_decrypt              |       2774      76.24  19.53 |        37      76.24 100.00   19.53 |       2737       0.00   0.00    0.00 |
  add_90                 |         37      76.24  19.53 |        37      76.24 100.00   19.53 |          0       0.00   0.00    0.00 |
   U0                    |         37      76.24  19.53 |        37      76.24 100.00   19.53 |          0       0.00   0.00    0.00 |
 dd_c                    |         75       0.00   0.00 |         0       0.00   0.00    0.00 |         75       0.00   0.00    0.00 |
 dd_d                    |        126      59.47  15.23 |        30      59.47 100.00   15.23 |         96       0.00   0.00    0.00 |
  add_90                 |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
   U0                    |         30      59.47  15.23 |        30      59.47 100.00   15.23 |          0       0.00   0.00    0.00 |
1
```

SEE ALSO

set_attribute(2)
set_max_lvth_percentage(2)
set_threshold_voltage_group_type(2)

report_timing

Displays timing information about a design.

SYNTAX

status **report_timing**
[-to *to_list*]
| -rise_to *rise_to_list*
| -fall_to *fall_to_list*]
[-from *from_list*
| -rise_from *rise_from_list*
| -fall_from *fall_from_list*]
[-through *through_list*]
[-rise_through *rise_through_list*]
[-fall_through *fall_through_list*]
[-exclude *exclude_list*
| -rise_exclude *rise_exclude_list*
| -fall_exclude *fall_exclude_list*]
[-path_type short | full | full_clock | full_clock_expanded | only | end | end_detailed]
[-delay_type min | min_rise | min_fall | max | max_rise | max_fall]
[-report_by design | mode | corner | scenario | group]
[-sort_by group | slack]
[-nworst *paths_per_endpoint*]
[-max_paths *max_path_count*]
[-slack_lesser_than *lower_slack_limit*]
[-pba_mode none | path | exhaustive]
[-start_end_pair]
[-significant_digits *digits*]
[-nosplit]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-groups *group_list*]
[-nets]
[-physical]
[-attributes]
[-input_pins]
[-transition_time]
[-capacitance]
[-process]
[-voltage]
[-temperature]
[-derate]
[-crosstalk_delta]
[-variation]
[-exception *exception_type*]
[-skip_transparency_window]

[-include_hierarchical_pins]

[*timing_path_list*]

Data Types

to_list list
rise_to_list list
fall_to_list list
from_list list
rise_from_list list
fall_from_list list
through_list list
rise_through_list list
fall_through_list list
exclude_list list
rise_exclude_list list
fall_exclude_list list
paths_per_endpoint integer
max_path_count integer
lower_slack_limit float
digits integer
mode_list list
corner_list list
scenario_list list
group_list list
timing_path_list collection

ARGUMENTS

-to *to_list*

Reports only the paths that connect to the named pins, ports, cells, or clocks. If you do not specify this option, the command reports the longest path to an output port if the design has no timing constraints. If the design has timing constraints, the command reports the path with the worst slack within each path group if **-group** is not present. If **-group** is given, the command reports the path with the worst slack within the group specified by **-group**.

-rise_to *rise_to_list*

Reports only the paths with a rising edge at the endpoint that connect to the named pins, ports, cells, or clocks. If a clock object is specified, this option selects endpoints clocked by the named clock, if the capturing edge is the rising edge of the clock at the clock source. The command considers logical inversions along the clock path.

-fall_to *fall_to_list*

Reports only the paths with a falling edge at the endpoint that connect to the named pins, ports, cells, or clocks. If a clock object is specified, this option selects endpoints clocked by the named clock, if the capturing edge is the falling edge of the clock at the clock source. The command considers logical inversions along the clock path.

-from *from_list*

Reports only the paths that originate from the named pins, ports, cells, or clocks. If you do not specify this option, the command reports the longest path to an output port if the design has no timing constraints. If the design has timing constraints, the command reports the path with the worst slack within each path group if **-group** is not present. If **-group** is given, the command

reports the path with the worst slack within the group specified by **-group**.

-rise_from ***rise_from_list***

Reports only the paths with a rising edge from the named pins, ports, cells, or clocks. If a clock object is specified, this option selects startpoints clocked by the named clock, if the paths are launched by the rising edge of the clock at the clock source. The command considers logical inversions along the clock path.

-fall_from ***fall_from_list***

Reports only the paths with a falling edge from the named pins, ports, cells, or clocks. If a clock object is specified, this option selects startpoints clocked by the named clock, if the paths are launched by the falling edge of the clock at the clock source. The command considers logical inversions along the clock path.

-through ***through_list***

Reports only paths that pass through the named pins, ports, cells, or nets. If you specify only one **-through** option, the tool reports only the paths that travel through one or more of the objects in the list. You can specify **-through** multiple times with one **report_timing** command. See the DESCRIPTION section for more information.

-rise_through ***rise_through_list***

Reports only paths that pass through the named pins, ports, cells, or nets and have a rising transition at those objects. If you specify only one **-through** option, the tool reports only the paths that travel through one or more of the objects in the list. You can specify **-rise_through** multiple times with one **report_timing** command. See the DESCRIPTION section for more information.

-fall_through ***fall_through_list***

Reports only paths that pass through the named pins, ports, cells, or nets and have a falling transition at those objects. If you specify only one **-through** option, the tool reports only the paths that travel through one or more of the objects in the list. You can specify **-fall_through** multiple times with one **report_timing** command. See the DESCRIPTION section for more information.

-exclude ***exclude_list***

Prevents the reporting of data paths that contain the specified pins, ports, nets, and cell instances. Reporting excludes all data paths from, through, or to the named pins, ports, nets, and cell instances. If a cell instance is specified, all pins of the cell are excluded. The **-exclude** option

o Takes precedence over the **-from**, **-through**, and **-to** options.

o Is exclusive with the **-rise_exclude** and **-fall_exclude** options.

o Does not apply to borrowing path from the **-trace_latch_borrow** option or clock path from the **-path full_clock** or **full_clock_expanded** option.

o Does not apply to clock pins.

Note that this option is intended for use in conjunction with **-from**, **-through** and **-to** options and can lead to long runtime when used on its own, i.e. without other such topological options.

-rise_exclude ***rise_exclude_list***

Same as the **-exclude** option, but applies only to paths rising at the named pins, ports, nets, and cell instances.

-fall_exclude ***fall_exclude_list***

Same as the **-exclude** option, but applies only to paths falling at the named pins, ports, nets, and cell instances.

-path_type short | **full** | **full_clock** | **full_clock_expanded** | **only** | **end** | **end_detailed**

Specifies how to display the timing path. By default, the full path is displayed. If **full_clock** is specified, the report is similar to the one generated by **full** but includes the full clock paths for propagated clocks. If **full_clock_expanded** is specified, the report is

similar to the one generated by **full_clock** but incudes full clock paths from the primary clock to the related generated clock source. If you specify **short**, only startpoints and endpoints are displayed. If you specify **only**, only the path is displayed without the accompanying required-time and slack calculation. If you specify **end**, the report has a column format that shows one line for each path, showing only the endpoint name, delay, required-time, slack, and scenario name. If you specify **end_detailed**, the report is similar to the one generated by **end**, but with additional columns showing the pathgroup name and the number of logic levels.

-delay_type min | min_rise | min_fall | max | max_rise | max_fall

Specifies the path type at the endpoint. The default is **max**.

-report_by design | mode | corner | scenario | group

Specifies the grouping criteria for reporting paths. The default is **design**. It is an error to give both the **-report_by** and **-sort_by** options.

-sort_by group | slack

Specifies the sorting criteria for reporting paths, group or slack. It is an error to give both the **-report_by** and **-sort_by** options. This option is deprecated in favor of the **-report_by** option.

-nworst *paths_per_endpoint*

Specifies the number of paths to report per endpoint. The default value is 1.

-max_paths *max_path_count*

Specifies the number of paths to report. The default value is 1. Depending of the value of the **-report_by** option, the max_paths value is applied to each path group, scenario, mode, or corner, or to the entire design.

-slack_lesser_than *lower_slack_limit*

Reports only those paths with a slack less than **lower_slack_limit**. When you specify this option, there might be an insufficient number of paths to satisfy the **-max_paths** and **-nworst** options. If this option is specified, unconstrained paths will not be returned, as unconstrained paths do not have a defined slack.

-pba_mode none | path | exhaustive

Controls path-based analysis with the following modes:

- **none** (the default) - Path-based analysis is not applied.
- **path** - Path-based analysis is applied to paths after they have been gathered.
- **exhaustive** - An exhaustive path-based analysis path search algorithm is applied to determine the worst path-based analysis path set in the design. If you use **exhaustive**, you cannot use the **-start_end_pair** option.

-start_end_pair

Reports paths for each pair of startpoint and endpoint based on connectivity. This option can lead to long runtime and can lead to generating a huge number of paths depending on the design. By default, this option searches only for paths that are violating. This default can be changed by having an explicit -slack_lesser_than option. The options that do not work with this option are -nworst, -max_paths, -report_by. Unlike other options of the report_timing command, this option causes the paths reported to no longer be sorted based on slack. Instead, paths are arranged based on the endpoint with those sharing the same endpoint appearing next to one another. The maximum number of paths reported is limited to 2000000.

-significant_digits *digits*

Specifies the number of significant digits (digits to the right of the decimal point) to report. Allowed values are from 0 through 13. The default is 2. Using this option overrides the value set by the **shell.common.report_default_significant_digits** application option.

-nosplit

Writes out wide report lines, even if the line length is greater than 80 columns. Use this option to simplify post-processing of the output files or when comparing the output files with the UNIX **diff** command. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

-modes *mode_list*

Specifies the modes for which to generate the timing report. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option will be included in the report. If this option is not given, the report will include scenarios of all modes. If none of the **-modes**, **-corners**, or **-scenarios** options are given, the report will include every active scenario of the design. It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to generate the timing report. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option will be included in the report. If this option is not given, the command reports every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios for which to generate the timing report. Each of the specified scenarios is included in the report. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the report will include every scenario of the design.

-groups *group_list*

Specify the list of path group names for reporting paths. Default is all path groups. If this option is specified, unconstrained paths will not be returned, as unconstrained paths do not belong to any path group.

-nets

Print nets.

-physical

Print pin XY locations.

-attributes

Print attributes of each pin.

-input_pins

Print data for input pins.

-transition_time

Print transition time of each pin.

-capacitance

Print capacitance of each load pin.

-process

Print process number and process label (if any) of each leaf pin.

-voltage

Print voltage of each leaf pin.

-temperature

Print temperature of each leaf pin.

-derate

Print timing derates of each leaf pin.

-crosstalk_delta

Print SI delta-delays of each load pin. This option is effective only if SI analysis is enabled.

-variation

Reports parametric on-chip variation (POCV) information by expanding column "Incr" to columns "Mean", "Sensit", "Corner" and "Value", and expanding column "Path" to columns "Mean", "Sensit" and "Value". This option works only when POCV analysis is enabled.

-exception *exception_type*

Prints user-specified timing exceptions, namely false paths, multicycle paths, and minimum and maximum delays, that are satisfied per timing path being reported. Only "dominant" exception type is acceptable, which is to report the dominant exception constraint used for the reported path.

Note: The additional analysis required per path with the -exceptions option is not trivial. Therefore, using a report_timing command with the -exceptions option is expected to execute slower than the exact same command without this option. The -exceptions option does not work with -path_type short/end/end_detailed option and forces the -input_pins option.

-skip_transparency_window

Skip printing the detailed transparency window.

-include_hierarchical_pins

Print the hierarchical pins.

timing_path_list

Specifies the collection of timing paths to report. This option is mutually exclusive with options that control the selection of paths to report and is only compatible with options which control the formatting of the report.

DESCRIPTION

The **report_timing** command generates a report that contains timing information for the current design. By default, the **report_timing** command reports the single worst setup path for each of the scenarios specified by the **-modes**, **-corners**, or **-scenarios** options. If none of these options are specified, the command will report the single worst path over all scenarios of the design.

Command options let you specify the number of paths reported, the types of paths reported, and the amount of detail included in the report. You can restrict the scope of paths reported by startpoints, endpoints, and intermediate points by using the **-from**, **-to**, and **-through** options. To restrict the report by slack or clock group, use the **-slack_lesser_than** and **-group** options.

If the **time.report_unconstrained_paths** application option is set to **true**, the command will search for unconstrained paths to endpoints for which constrained paths cannot be found. Searching for unconstrained paths can cause significantly longer runtimes.

The report actually consists of a series of sub-reports, each containing up to **max_paths** paths, sorted by slack. The value of the **-report_by** option controls the contents of each sub-report as follows:

design \(em There will be a single sub-report for the entire design, containing the worst paths across all pathgroups of all scenarios. This is the default.

group \(em There will be a sub-report (with up to **max_paths** paths) for each pathgroup of each specified scenario. (This is equivalent to the old default **-sort_by group** option.)

scenario \(em There will be a sub-report for each scenario, containing the worst paths across all pathgroups. (This is equivalent to the old **-sort_by slack** option.)

mode \(em There will be a sub-report for each mode, containing the worst paths across all pathgroups of all scenarios of that mode.

corner \(em There will be a sub-report for each corner, containing the worst paths across all pathgroups of all scenarios of that corner.

The command can also report on a collection of timing paths which were retrieved using the **get_timing_paths** command.

Multicorner-Multimode Support

By default, this command works on all active scenarios. To specify different scenarios, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example uses multiple **-through** options in a single command to specify paths that traverse multiple points in the design. The command reports paths beginning at A1, passing through B1, then through C1, and ending at D1:

```
prompt> report_timing -from A1 -through B1 -through C1 -to D1
```

The following example allows multiple possible paths by specifying multiple objects within one **-through** option. The path can pass through any of the objects. The command specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
prompt> report_timing -from A1 -through {B1 B2} -through {C1 C2} -to D1
```

The following example uses the **-report_by design** option to report the single worst path over the entire design:

```
prompt> report_timing -report_by design -scenarios [all_scenarios] -max_paths 1
```

The following example shows how a pre-existing collection of paths can be printed:

```
prompt> set paths [get_timing_paths -report_by corner -corner c23 -max_paths 100]
prompt> report_timing $paths
```

If you specify the **-pba_mode exhaustive** option, path-based analysis is used during the path search, and the worst recalculated paths meeting your criteria are returned. This option requires that a path search be performed to ensure that the paths being returned truly are the worst paths. The additional runtime required for this option depends on the amount of timing improvement resulting from path-based analysis. As the timing improvement from path-based analysis increases, the runtime for the path search increases. Large **-nworst** values can significantly increase the time needed for the search.

To see the worst path in the design with path-based analysis applied:

```
prompt> report_timing -pba_mode exhaustive
```

To report all endpoints in the design which fail after considering path-based analysis:

```
prompt> report_timing -pba_mode exhaustive -slack_lesser_than 0 -max_paths 1000
```

SEE ALSO

current_corner(2)
current_mode(2)
current_scenario(2)
get_timing_paths(2)
set_scenario_status(2)
time.report_unconstrained_paths(3)

report_timing_derate

Reports derate factors annotated on the current design.

SYNTAX

int **report_timing_derate**
[-corners *corner_list*]
[-include_inherited]
[-aocvm_guardband]
[-pocvm_guardband]
[-pocvm_coefficient_scale_factor]
[-pocvm_subtract_sigma_factor_from_nominal]
[-increment]
[-significant_digits *digits*]
[-nosplit]
[*object_list*]

Data Types

corner_list list
digits int
object_list list

ARGUMENTS

-corners *corner_list*

Reports timing derates for all the corners specified in *corner_list*. The default is to only report the current corner.

-include_inherited

Indicates that the derate factors reported on each object should also include what object they inherit their value from in the design.

-aocvm_guardband

Indicates that only the AOCVM guard band derate factors should be reported.

-pocvm_guardband

Indicates that only the POCVM guardband derate factors should be reported.

-pocvm_coefficient_scale_factor

Indicates that only the POCVM coefficient scale factors should be reported.

-pocvm_subtract_sigma_factor_from_nominal

Indicates that only the POCVM subtract sigma factors should be reported.

-increment

Indicates that only incremental derate factors should be reported. The option -increment and -aocvm_guardband/-pocvm_guardband are mutually exclusive.

-significant_digits *digits*

Specifies the number of digits after the decimal point to be displayed for values in the report. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option. Use this option if you want to override the default.

-nosplit

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line-splitting and facilitates writing scripts to extract information from the report output.

object_list

Specifies a list of cells, lib_cells in the current design.

DESCRIPTION

Reports the derate factors either on the design or specific instances (cells or library cells) contained in the design. If this command is successful a Boolean of true or 1 is returned; otherwise, it returns a false or 0.

If this command is called with no arguments, every object where a timing derate value has been applied it listed. The inheritance information is not displayed.

If this command is called with the *-include_inherited* Boolean argument, the derate report for each instance also contains the name of the object from which the derate value is inherited. Examples of this would be a leaf cell inheriting cell derate factors set on the hierarchical cell to which it belongs, or a leaf cell inheriting a derate factor set on the library cell of which it is an instantiation.

Because net derating factors are global, they are displayed only under the top hierarchical design (the current_design). With the support of dynamic derates, the report shows 2 separate lines for static and dynamic net derates.

If the command is called with the *object_list* option specified, derate reports are only issued for the instances specified in the object list.

Multicorner-Multimode Support

By default, this command works on the current corner. To specify a different corner, use the **-corners** option.

EXAMPLES

In the following example derate factors have only been set globally (on the design), therefore only global derate factors are reported.

prompt> **set_timing_derate -data -early 0.92**

```
prompt> set_timing_derate -data  -late  1.14
prompt> set_timing_derate -clock -early 0.75
prompt> set_timing_derate -clock -late  1.26
prompt> report_timing_derate

****************************************
Report : timing_derate
Design : core_cpt
****************************************

Timing derate for corner C2

                    ----- Clock -----              ----- Data -----
                  Rise            Fall            Rise            Fall
               Early    Late    Early    Late    Early    Late    Early    Late

Design : core_cpt
 Net delay static     1.00     1.00     1.00     1.00     1.00     1.00     1.00     1.00
 Net delay dynamic    1.00     1.00     1.00     1.00     1.00     1.00     1.00     1.00
 Cell delay           0.75     1.26     0.75     1.26     0.92     1.14     0.92     1.14

1
```

In the following example derate factors have been set on the design named 'core_cpt'. More restrictive derates have been set on the hierarchical block name 'H1'. The derates that apply to the hierarchical cell named 'H1/ND1' are reported.

```
prompt> set_timing_derate -early 0.95
prompt> set_timing_derate -late  1.06
prompt> set_timing_derate -early 0.82 [get_cells H1]
prompt> set_timing_derate -late  1.12 [get_cells H1]
prompt> report_timing_derate -include_inherited [get_cells H1/u5]

****************************************
Report : timing_derate
Design : core_cpt
****************************************

Timing derate for corner C2

                    ----- Clock -----              ----- Data -----
                  Rise            Fall            Rise            Fall
               Early    Late    Early    Late    Early    Late    Early    Late

Cell (Leaf) : H1/u5
 Cell delay           0.82     1.12     0.82     1.12     0.82     1.12     0.82     1.12
 Inherited            H1       H1       H1       H1       H1       H1       H1       H1
```

Following example shows the report with incremental derates that have been set on the design named 'core_cpt'. Both static and dynamic net derate factors are set.

```
prompt> set_timing_derate -increment -early -0.05
prompt> set_timing_derate -increment -late   0.05
prompt> set_timing_derate -increment -early -0.03 [get_cells u1]
prompt> set_timing_derate -increment -late   0.03 [get_cells u1]
prompt> report_timing_derate -increment
```

```
****************************************
Report : timing_derate
Design : core_cpt
****************************************

Timing derate for corner C2

                    ----- Clock -----              ----- Data -----
                  Rise            Fall            Rise            Fall
                Early   Late    Early   Late    Early   Late    Early   Late

Design : core_cpt
  Net delay static     -0.05    0.05    -0.05    0.05    -0.05    0.05    -0.05    0.05
  Net delay dynamic    -0.05    0.05    -0.05    0.05    -0.05    0.05    -0.05    0.05
  Cell delay           -0.05    0.05    -0.05    0.05    -0.05    0.05    -0.05    0.05
  Cell check            0.00    0.00    0.00     0.00    0.00     0.00    0.00     0.00

Cell (Leaf) : u1
  Cell delay           -0.03    0.03    -0.03    0.03    -0.03    0.03    -0.03    0.03
  Cell check            0.00    0.00    0.00     0.00    0.00     0.00    0.00     0.00
```

SEE ALSO

set_timing_derate(2)
reset_timing_derate(2)

report_topological_constraints

Generates a report of the topological pin feedthrough constraint information.

SYNTAX

status **report_topological_constraints**
topological_constraints | -all

topological_constraints collection

ARGUMENTS

topological_constraints

Specifies a list of topological pin feedthrough constraints from the design that are to be reported. Each topological pin feedthrough constraint name in the *topological_constraints* must exist in the design.

You must specify the *topological_constraints* or **-all**. The *topological_constraints* and **-all** options are mutually exclusive.

-all

Reports all the topological pin feedthrough constraints in the design. The *topological_constraints* and **-all** options are mutually exclusive.

DESCRIPTION

This command displays information about the topological pin feedthrough constraints in the current design.

EXAMPLE

The following example reports the topological pin feedthrough constraint TPF_CONSTRAINT_0

```
prompt> report_topological_constraints TPF_CONSTRAINT_0
TPF Constraint     Description
-------------------------------------------------------------------------------
TPF_CONSTRAINT_0   Owner Type          blk_net
                   Owner Name          pad[15]
                   Start Type          blk_inst
```

```
                Start Name          DFTC_4
                End Type            blk_inst
                End Name            sdram_DQ_iopad_14
                Start Sides         --
                End Sides           --
                Start Offset        --
                End Offset          --
                Start Offset Range    --
                End Offset Range      --
                Start Layers        --
                End Layers          --
1
```

The following example reports all the topological pin feedthrough constraints in the current design

```
prompt> report_topological_constraints -all
TPF Constraint     Description
-------------------------------------------------------------------------------
TPF_CONSTRAINT_0    Owner Type          blk_net
                Owner Name          pad[15]
                Start Type          blk_inst
                Start Name          DFTC_4
                End Type            blk_inst
                End Name            sdram_DQ_iopad_14
                Start Sides         --
                End Sides           --
                Start Offset        --
                End Offset          --
                Start Offset Range    --
                End Offset Range      --
                Start Layers        --
                End Layers          --

TPF_CONSTRAINT_1    Owner Type          blk_net
                Owner Name          pad[14]
                Start Type          blk_inst
                Start Name          DFTC_4
                End Type            blk_inst
                End Name            sdram_DQ_iopad_14
                Start Sides         --
                End Sides           --
                Start Offset        --
                End Offset          --
                Start Offset Range    --
                End Offset Range      --
                Start Layers        --
                End Layers          --

TPF_CONSTRAINT_27   Owner Type          blk_net
                Owner Name          pad[15]
                Start Type          blk_inst
                Start Name          DFTC_4
                End Type            voltage_area_region
                End Name            VOLTAGE_AREA_SHAPE_0
                Start Sides         {1 2 3 4}
                End Sides           {1 2 3 4}
                Start Offset        1.8
```

```
            End Offset          10
            Start Offset Range    --
            End Offset Range      --
            Start Layers         --
            End Layers           --

    TPF_CONSTRAINT_28   Owner Type          blk_net
            Owner Name           pad[15]
            Start Type          blk_inst
            Start Name           sdram_DQ_iopad_14
            End Type            voltage_area_region
            End Name             VOLTAGE_AREA_SHAPE_0
            Start Sides         {1 2 3 4}
            End Sides            {1 2 3 4}
            Start Offset        10
            End Offset          10
            Start Offset Range    2
            End Offset Range      2
            Start Layers         --
            End Layers           --

    TPF_CONSTRAINT_29   Owner Type          blk_net
            Owner Name           pad[15]
            Start Type          blk_inst
            Start Name           DFTC_4
            End Type            voltage_area
            End Name             DEFAULT_VA
            Start Sides         {1 2 3 4}
            End Sides            {1 2 3 4}
            Start Offset        3.4
            End Offset          10
            Start Offset Range    2
            End Offset Range      2
            Start Layers        {METAL METAL2}
            End Layers           {METAL4 METAL6}
    1
```

SEE ALSO

create_topological_constraint(2)
get_topological_constraints(2)
remove_topological_constraints(2)

report_topology_plans

Reports topology plans in the current block.

SYNTAX

int **report_topology_plans**
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*topology_plan_list*]

Data Types

digits int
topology_plan_list collection

ARGUMENTS

-verbose

Displays verbose topology plan information.

-nosplit

Retains long lines and does not split lines if column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point to report. Allowed values are 0-13. The default is determined by the **shell.report_default_significant_digits** application option, whose default is 2. Use this option to override the default.

topology_plan_list

Specifies a list of topology plans to report. The list can contain topology plan names, patterns, or collections. A collection can be specified by using the **get_topology_plans** command.

DESCRIPTION

The **report_topology_plans** command generates a report about the topology plans in the design. The report includes the topology plan name and a description of the topology plan. If **-verbose** is specified, the report also includes additional attributes and a list of constraint groups the topology plan is a member of.

If *topology_plan_list* is specified, those topology plans are reported. If it is not specified, then all topology plans in the design are reported.

EXAMPLES

The following example reports information for the topology_plan named "plan0".

```
prompt> report_topology_plans plan0
Topology Plan     Description
-------------------------------------------------------------------------------
plan0           comment: Plan for decoder sys 3
                stats: 2 nodes, 1 edges, 2 repeaters
                net_estimation_rule: nerule1
                objects: 1 pins, 1 ports, 2 nets, 1 bundles, 1 supernets
                nodes:
                    plan0/p0_node1
                            is_reference: false
                            num_edges: 1
                            num_driving_edges: 1
                            objects: 7 cells

                    plan0/p0_node0
                            is_reference: false
                            num_edges: 1
                            num_driving_edges: 0
                            objects: 2 cells, 1 pins, 1 ports, 2 voltage_area_shapes

                edges:
                    plan0/e0_dat
                            start_node: plan0/p0_node0
                            end_node: plan0/p0_node1
                            net_estimation_rule: nerule5_x
                            objects: 1 pins, 1 ports, 2 nets, 1 bundles, 1 supernets
                            repeaters:
                                    plan0/e0_dat/rptr0
                                        type: repeater

                                    plan0/e0_dat/rptr1
                                        type: repeater

1
```

SEE ALSO

create_topology_node(2)
create_topology_plan(2)
get_topology_nodes(2)
get_topology_plans(2)
remove_topology_nodes(2)

remove_topology_plans(2)

report_trace

Produce a report of performance profile metrics collected for traced commands.

SYNTAX

report_trace [-start|-stop|resume]
[-profile profile_type]
[-command command_name]
[-cumulative count]
[-top count]
[-quiet]

string *profile_type*
string *command_name*
int *count*

ARGUMENTS

-start

This option is mutually exclusive with -stop, -resume, -profile, -top, -cumulative, -command. The option starts the collection of performance profile statics for each traced command invocation. If metric collection had previously been stopped with -stop, then previously collected data are deleted before collection starts.

-stop

This option is mutually exclusive with -start,-resume, -profile, -top, -cumulative, -command. The option stops the collection of performance profile statics for traced command invocations.

-resume

This option is mutually exclusive with -start, -stop, -profile, -top, -cumulative, -command. The option resume the collection of performance profile statics for traced command invocations. Previous gathered statics remain and newly collected metrics are added to pre-existing data. This option may be used after -stop to resume data collection without deleting previously collected data.

-profile *profile_type*

This option is mutually exclusive with -start, -stop, and -resume. This option selects the profile metrics that are included in the report. Allowed valuesA valid value is a list of permitted values {memory, cpu, time, count}, or the value all to select all metrics. If -profile is given, then the report output contains performance profile data of given types only. If the option is omitted, then it defaults to all.

-cumulative *count*

This optional option is mutually exclusive with -start, -stop, -resume, and -command. This option limits the number of commands reported for the top resource consumer accumulated over all invocations. However, all commands that share the same usage metric as the minimum consumer included in the count will be shown. If the option is omitted, the report output contains the top 10 consumers.

-top *count*

This optional option is mutually exclusive with -start, -stop, -resume and -command. This option limits the number of commands reported for the top resource consumer per single invocation. However, all commands that share the same usage metric as the minimum consumer included in the count will be shown. If the option is omitted, the report output contains the top 10 consumers.

-command *command_name*

This option is mutually exclusive with -start, -stop, -resume, -top and -cumulative. This option causes a performance profile report to be displayed for the given command. Cumulative and top single invocation metrics are shown. If the command was not invoked, then a message with this information is displayed instead of a report.

-quiet

If given, this option causes the command to ignore error conditions such as an attempt to start profile gathering when it has already been started or reporting on a non-existent command. The command will exit quietly when it encounters an error condition.

DESCRIPTION

The **report_trace** command collects performance profile metrics on all traced command invocations since the last -start. Stopping the gathering is not necessary to produce a report. When performance profile collection is started with report_trace -start, each traced command will be measured for the available performance metrics: memory, CPU, wallclock time, and call count. The accumulated profile data can be reported at any time by calling this command

Stopping the metric gathering with -stop freezes the gathered metrics until the next -start. When gathering is restarted, all previously gathered metrics are deleted. Alternatively, metric collection may be restarted without deleting previously collected data using -resume.

The **report_trace** with no options will report the top 10 consumers in both cumulative and single call metrics for all profile types collected thus far.

EXAMPLES

The following example starts the collection of performance profile metrics.

```
prompt> report_trace -start
on
```

The following example produces a report of top 10 commands with most usage for all profile types for both cumulative and individual call consumption. 10 is the default number reported for -top and -cumulative when these options are omitted.

```
prompt> report_trace
on
```

The following example produces a report of cumulative and the single most expensive invocation for each profile type for the command update_timing.

```
prompt> report_trace -command update_timing
on
```

The following example produces a report of top 20 consumers for cumulative and single invocation collected thus far.

```
prompt> report_trace -top 20 -cumulative 20
on
```

The following example produces a report of top 5 cumulative consumption of memory.

```
prompt> report_trace -profile memory -cumulative 5
on
```

The following example produces a report of top 15 commands with most invocations.

```
prompt> report_trace -profile count -cumulative 15
on
```

The following example produces a report of top 10 commands with most cpu and wallclock time usage for both cumulative and individual call consumption. 10 is the default number reported for -top and -cumulative when these options are omitted.

```
prompt> report_trace -profile {cpu time}
on
```

SEE ALSO

set_trace_option(2)
get_trace_option(2)
log_trace(2)
annotate_trace(2)

report_track_constraints

The command outputs a report describing the track constraints set for the current block.

SYNTAX

int **report_track_constraints**

DESCRIPTION

The report_track_constraints command takes no arguments. It displays information on the track constraints defined in the current block. Labels that exceed the defined column width in the report are automatically condensed as seen in the below example with the label "thisIsAReallyLongLabelString" being reported as "thisIsAReallyLon...".

EXAMPLES

The following commands set and remove track constraints and then show the resulting track constraint report.

```
prompt> set_track_constraint -block [current_block] -layer M1 \
  -label A -grid 1 -offset 1 -mask mask_one -origin block
prompt> set_track_constraint -block [current_block] -layer M2 \
  -label B -grid 1 -offset {0 0.5} -mask {mask_one mask_two} -origin block
prompt> set_track_constraint -block [current_block] -layer M2 \
  -label C -grid 4.5 -offset 1 -mask mask_one -origin block \
  -track_direction vertical
prompt> set_track_constraint -block [current_block] -layer M3 \
  -label thisIsAReallyLongLabelString -grid 1 -offset {0 0.1 0.9} \
  -mask mask_two -origin block -track_direction horizontal
prompt> remove_track_constraint -block [current_block] -layer M1 \
  -mask {mask_one mask_two}
prompt> remove_track_constraint -block [current_block] -layer M2 \
  -mask mask_one -track_direction vertical
prompt> report_track_constraints
Track Constraint Report
   Layer: M1
   Label          Constraint            Origin   Grid  Offset

   Layer: M2
   Label          Constraint            Origin   Grid  Offset
   B            {Horizontal, mask_one}   block    1    {0 0.5}
```

```
B           {Vertical, mask_one}     block    1    {0 0.5}
B           {Vertical, mask_two}     block    1    {0 0.5}

Layer: M3
Label       Constraint               Origin   Grid  Offset
thisIsAReallyLon... {Vertical, mask_one}     block    1    {0 0.1 0.9}

Layer: M4
Label       Constraint               Origin   Grid  Offset

Layer: M5
Label       Constraint               Origin   Grid  Offset

Layer: M6
Label       Constraint               Origin   Grid  Offset

Layer: M7
Label       Constraint               Origin   Grid  Offset
```

SEE ALSO

set_track_constraint(2)
remove_track_constraint(2)

report_tracks

Reports the routing tracks for a specified layer or for all layers.

SYNTAX

int **report_tracks**
[-nosplit]
[-significant_digits *digits*]
[-layer *layer*]
[-dir X | Y]

Data Types

digits int
layer string

ARGUMENTS

-nosplit

Does not split lines if column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **shell.commmon.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default.

-layer *layer*

Specifies the routing layer to use the routing tracks. You can use layer name, layer number, or a collection containing one layer object.

The default is to report the routing tracks on all layers.

-dir X | Y

Specifies the direction how routing tracks are placed. The valid values are **X** and **Y**; specify either. The default is to report routing tracks in both direction.

DESCRIPTION

This command reports a group of routing tracks for the routing layer.

EXAMPLES

The following example reports routing tracks on the floorplan.

```
prompt> report_tracks
Layer        Direction    Start       Tracks   Pitch       Attr
-----------------------------------------------------------------------

m2             Y       233.720      2861     0.560     default
m2             X       233.720      2649     0.560     default
m3             X       233.720      2649     0.560     default
m3             Y       233.720      2861     0.560     default
m2             Y       233.720      2861     0.560     default
m4             Y       233.720      2861     0.560     default
m4             X       233.720      2649     0.560     default
m3             X       233.720      2649     0.560     default
m5             X       233.720      2649     0.560     default
m5             Y       233.720      2861     0.560     default
m4             Y       233.720      2861     0.560     default
m6             Y       233.720      2861     0.560     default
m6             X       232.600      1327     1.120     default
m5             X       232.600      1327     1.120     non-reserved, width=0.5
m1             X       20.000       10       2.100     reserved, width=0.8
1
prompt> report_tracks -layer m3 -dir Y
****************************************
Report track
Design : design
 ...

****************************************
Layer        Direction    Start       Tracks   Pitch       Attr
-----------------------------------------------------------------------
Attributes :
      usr : User defined
      rt  : Route66 defined
      def : DEF defined

m3             Y       233.720      2861     0.560
1
```

SEE ALSO

create_track(2)
remove_tracks(2)
get_tracks(2)

report_transformed_registers

Reports register transformations during compile

SYNTAX

status **report_transformed_registers**
[-nosplit]
[-constants]
[-unloaded]
[-shiftReg]
[-multibit]
[-merged]
[-inverted]
[-replicated]
[-summary]

Data Types

ARGUMENTS

-nosplit

Does not split lines if the column overflow.

-constants

Reports constant regsiters in the design. The report will list constant registers that are optimized as well as constant register that are not removed due to user constraints.

-unloaded

Reports unloaded regsiters in the design. The report will list unloaded registers that are optimized as well as unloaded register that are not removed due to user constraints.

-shiftReg

Reports shift registers inferred by the tool.

-multibit

Reports multibit registers inferred by the tool.

-merged

Report registers that are optimized away by transferring loads to a logically equivalent register.

-inverted

Reports phase inverted registers. The report only includes register whose final state is phase inverted compared to its original state.

-replicated

Report registers that are duplicated to meeting optimization objective.

-summary

Gives total count of each individual transformations.

DESCRIPTION

This command reports the registers undergoing transformations during compile. If no option is specified then all the transformations are reported by default. Any of the above options can be independently specified to report that specific optimization.

EXAMPLES

prompt> **report_transformed_registers -constants**

prompt> **report_transformed_registers**

```
****************************************
Report : report_registers
 ...
****************************************
Attributes:
C0p - constant 0 register preserved
C1p - constant 1 register preserved
C0r - constant 0 register removed
C1r - constant 1 register removed
ulp - preserved unloaded register
ulr - removed unloaded register
mrg - merged register
srh - shift-register head
srf - shift-register flop
mb  - multibit
rep - replicated register
inv - output inverted register

Register                                        Optimization
------------------------------------------------------------------------------ ------------
i_sub1/Stack_Mem_reg[6][0]                                  C0p
i_sub1/Stack_Mem_reg[7][1]                                  C1r
```

prompt> **report_transformed_registers -summary**

prompt> **report_transformed_registers**

```
****************************************
```

```
Report : report_registers
...
****************************************
Attributes:
C0p - constant 0 register preserved
C1p - constant 1 register preserved
C0r - constant 0 register removed
C1r - constant 1 register removed
ulp - preserved unloaded register
ulr - removed unloaded register
mrg - merged register
srh - shift-register head
srf - shift-register flop
mb  - multibit
rep - replicated register
inv - output inverted register

Register                                                  Count
--------------------------------------------------------------------------------- ----------
Constant Registers Deleted                                    2
Constant Registers Preserved                                  0
Unloaded Registers Deleted                                    0
Unloaded Registers Preserved                                  1
Shift Registers                                               0
Merged                                                        0
Multibit                                                      0
Inverted                                                      0
Replicated                                                    0
```

SEE ALSO

report_constant_registers(2)
report_unloaded_registers(2)

report_transitive_fanin

Reports logic in the transitive fanin of specified sinks.

SYNTAX

status **report_transitive_fanin**
-to *sink_list*
[-nosplit]

Data Types

sink_list list

ARGUMENTS

-to *sink_list*

Specifies a list of sink pins, ports, or nets in the design. The transitive fanin of each sink in the *sink_list* is reported. If a net is specified, the effect is the same as listing all driver pins on the net. This argument is required.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_transitive_fanin** command produces a report showing the transitive fanin of specified sink pins, ports, or nets in the design. A pin is considered to be in the transitive fanin of a sink if there is a path through combinational logic from the pin to that sink. The fanin report stops at the clock pins of registers (sequential cells). It is independent on timing graph.

Use the **report_transitive_fanout** command to see the fanout of a pin, port, or net.

If the *current_instance* is set, the report focuses on the fanin within the current instance. The report stops at the boundaries of the current instance, and paths outside the current instance are not reported. The report shows the hierarchical boundary pins on the current instance.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows the transitive fanin of two internal pins in the design:

```
prompt> report_transitive_fanin -to {u5/A u5/B}

****************************************
Report : transitive_fanin
 ...

****************************************

The fanin network of sink u5/A is as follows:

Driver Pin      Load Pin        Type      Sense
-------------------------------------------------------------
u2/Z            u5/A            (net arc)   same

Load Pin        Driver Pin      Type      Sense
-------------------------------------------------------------
u2/A            u2/Z            IVA       opposite

Driver Pin      Load Pin        Type      Sense
-------------------------------------------------------------
b               u2/A            (net arc)   opposite

The fanin network of sink u5/B is as follows:

Driver Pin      Load Pin        Type      Sense
-------------------------------------------------------------
c               u5/B            (net arc)   same
```

SEE ALSO

current_design(2)
current_instance(2)
report_clock(2)
report_timing(2)
report_transitive_fanout(2)

report_transitive_fanout

Reports logic in the transitive fanout of specified sources.

SYNTAX

status **report_transitive_fanout**
-from *source_list*
[-nosplit]

Data Types

source_list list

ARGUMENTS

-from *source_list*

Specifies a list of source pins, ports, and nets in the design. The transitive fanout of each source in the *source_list* is reported. If a net is specified, the effect is the same as listing all load pins on the net.

-nosplit

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

DESCRIPTION

The **report_transitive_fanout** command produces a report showing the transitive fanout of specified source pins or ports in the design. A pin is considered to be in the transitive fanout of a source if there is a path through combinational logic from the source to that pin. The fanout report stops at the inputs to registers (sequential cells). The source pins or ports are specified with **-from** *source_list*. It is independent on timing graph.

Use the **report_transitive_fanin** command to see the fanin of a pin, port, or net.

If the *current_instance* is set, the report focuses on the fanout within the current instance. The report stops at the boundaries of the current instance, and does not report paths outside the current instance. The report also shows the hierarchical boundary pins on the current instance.

Multicorner-Multimode Support

This command uses information from the current scenario only.

EXAMPLES

The following example shows the transitive fanout of two internal pins in the design:

```
prompt> report_transitive_fanout -from {ffa/Q ffb/Q}

****************************************
Report : transitive_fanout
 ...

****************************************

The fanout network of source ffa/Q is as follows:

Driver Pin      Load Pin        Type      Sense
-------------------------------------------------------------
ffa/Q           QA/A            (net arc)  same

Load Pin        Driver Pin      Type      Sense
-------------------------------------------------------------
QA/A            QA/Z            IV        opposite

Driver Pin      Load Pin        Type      Sense
-------------------------------------------------------------
QA/Z            QA              (net arc)  opposite

The fanout network of source ffb/Q is as follows:

Driver Pin      Load Pin        Type      Sense
-------------------------------------------------------------
ffb/Q           QB/A            (net arc)  same

Load Pin        Driver Pin      Type      Sense
-------------------------------------------------------------
QB/A            QB/Z            IV        opposite

Driver Pin      Load Pin        Type      Sense
-------------------------------------------------------------
QB/Z            QB              (net arc)  opposite
```

SEE ALSO

current_design(2)
current_instance(2)
report_transitive_fanin(2)

report_unbound

Reports unbound cell, via, via array, site row, site array, shape, track, layer, pin_guide, pin_blockage, routing_guide, routing_corridor_shape and via_def object types.

SYNTAX

int **report_unbound**
[-hierarchical]
[-cell]
[-via]
[-site_row]
[-site_array]
[-shape]
[-track]
[-layer]
[-pin_guide]
[-pin_blockage]
[-routing_guide]
[-routing_corridor_shape]
[-via_def]
[-verbose]

Data Types

No data types.

ARGUMENTS

-hierarchical

Reports unbound objects within the physical hierarchy.

-cell

Reports unbound cell instance objects. Specify **-hierarchical** to consider cell instances within the physical hierarchy. Specify **-verbose** to report cell instances with missing references.

-via

Reports unbound via and via array objects. Specify **-hierarchical** to report via and via arrays within the physical hierarchy. Specify **-verbose** to report via and via arrays with missing references.

-site_row

Reports unbound site row objects. Specify **-hierarchical** to report unbound site rows within the physical hierarchy. Specify **-**

verbose to report site rows with missing site definitions.

-site_array

Report unbound site array objects. Specify **-hierarchical** to report unbound site arrays within the physical hierarchy. Specify **-verbose** to report site arrays with missing site definitions.

-shape

Reports unbound layer along with shape object name in the command output.

-track

Reports unbound layer along with track object name in the command output.

-layer

Reports unbound layer for layer object.

-pin_guide

Reports unbound layer along with pin guide object name in the command output.

-pin_blockage

Reports unbound layer along with pin blockage object name in the command output.

-routing_guide

Reports unbound layer along with routing guide object name. If type of routing guide is access_preference then it will also be mentioned in the command output.

-routing_corridor_shape

Reports unbound layer along with routing corridor shape object name. Reports min_layer_name and max_layer_name in command output if layers are unbound.

-via_def

Reports unbound layer along with via_def object name. Reports upper, lower and cut layers for via_def object if they are unbound.

-verbose

Reports cell instances with missing references. If only this option is specified, the command reports cell instance objects with missing references. Specify **-verbose** and **-hierarchical** to report unbound cell instances within the physical hierarchy.

DESCRIPTION

This command displays information for unbound objects in the design. The object types considered are cell, via, via matrix, site row, site array, shape, track, layer, pin_guide, pin_blockage, routing_guide, routing_corridor_shape and via_def.

With **-verbose** option, for cell, via, via matrix, site row, site array objects, the command reports each unbound object and its reference name. for other supporting objects it reports the unbound layer with object name. With **-hierarchical** option, for cell, via, via matrix, site row, site array objects, the command reports objects inside physical hierarchy as well.

If none of the options are specified then for **-cell**, **-via**, **-site_row** and **-site_array**, the tool prints out all missing references of all these object types considered and the count of unbound objects per reference. For shape, track, layer, pin_guide, pin_blockage,

routing_guide, routing_corridor_shape and via_def tool prints out all the unbound layers with object name.

EXAMPLES

The following example reports unbound objects for cell, via, site row, site array, shape, track, layer, pin_guide, pin_blockage, routing_guide, routing_corridor_shape and via_def object types.

```
prompt> report_unbound
****************************************
Report : design
Design : test_logic1
Version: O-2018.06-SP2-BETA
Date   : Tue June 12 23:15:15 2018
****************************************
Error: Library cell 'AN2D0BWP7D5T16P96CPD'
  missing. 'u_ecore0/u_esub0/u_fcche/u_fcche_sif/U50' and '14491' other
  cell instance(s) which reference this lib cell are
  unbound. (CHUNB-001)
Error: Library cell 'AN2D16BWP7D5T16P96CPD'
  missing. 'u_ecore0/u_esub12/u_esub1/u_me/u_sme/u_sme_ctrl/U458'
  and '1' other cell instance(s) which reference this lib cell are
  unbound. (CHUNB-001)
Error: Library cell 'AN2D1BWP7D5T16P96CPD'
  missing. 'u_ecore0/u_esub0/u_fme/u_fme_cost_0/U1414' and '147' other
  cell instance(s) which reference this lib cell are
  unbound. (CHUNB-001)
Error: Site definition 'gaunit' is missing. 'DEFAULT_SA_1' and '2564'
  other site row(s) which reference this site definition are
  unbound. (CHUNB-003)
Error: Site definition 'gaunit' is missing. 'DEFAULT_SA' and '0' other
  site array(s) which reference this site definiton are
  unbound. (CHUNB-005)
Error: Via definition 'VIA12_1cut' is missing. 'VIA_S_0'and '2136340'
  other via(s) which reference this via definition are
  unbound. (CHUNB-007)
Error: Shape 'RECT_133_0' has unbound layer '133:0'. (CHUNB-009)
Error: Track 'TRACK_0' has unbound layer '31:0'. (CHUNB-010)
Error: Layer 'L1:124' is unbound layer. (CHUNB-011)
Error: Pin guide 'SNPS_PG_0' has unbound layer '300:0'. (CHUNB-012)
Error: Pin blockage 'SNPS_PB_0' has unbound layer '31:0'. (CHUNB-013)
Error: Routing guide 'RG_0' has unbound layer '31:0'. (CHUNB-014)
Error: Routing corridor shape 'CORRIDOR_SHAPE_0' has unbound layer '32:0'. (CHUNB-015)
Error: simple_via_def 'VIA12_FAT' has unbound layer(upper layer) '32:0'. (CHUNB-016)
Error: simple_via_def 'VIA12_FAT' has unbound layer(lower layer) '31:0'. (CHUNB-016)
Error: simple_via_def 'VIA12_FAT' has unbound layer(cut layer) '51:0'. (CHUNB-016)
Error: custom_via_def 'CUSTOM1' has unbound layer(upper layer) '32:0'. (CHUNB-016)
Error: custom_via_def 'CUSTOM1' has unbound layer(lower layer) '31:0'. (CHUNB-016)
Error: custom_via_def 'CUSTOM1' has unbound layer(cut layer) '51:0'. (CHUNB-016)
Error: through_silicon_via_def 'VIAB1' has unbound layer(upper layer) '13:0'. (CHUNB-016)
Error: through_silicon_via_def 'VIAB1' has unbound layer(lower layer) '11:0'. (CHUNB-016)
Error: through_silicon_via_def 'VIAB1' has unbound layer(cut layer) '12:0'. (CHUNB-016)
1
```

SEE ALSO

remove_objects(2)
remove_site_defs(2)
remove_via_defs(2)

report_units

Reports the currently specified input and output units.

SYNTAX

string **report_user_units**
[-nosplit]

ARGUMENTS

-nosplit

Retains long lines in the output, even if the lines are longer than 80 columns. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing scripts to extract information from the report output.

DESCRIPTION

The **report_user_units** command displays the currently specified input and output units.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example reports the user units for the current design.

```
prompt> report_user_units

****************************************
Report : user_units
 ...

****************************************
```

```
Input Units (set to default)
-------------------------------------------------
time                : 1.00ns
resistance          : 1.00kOhm
capacitance         : 1.00pF
voltage             : 1.00V
current             : 1.00mA
power               : 1.00mW

Output Units (set to default)
-------------------------------------------------
time                : 1.00ns
resistance          : 1.00kOhm
capacitance         : 1.00pF
voltage             : 1.00V
current             : 1.00mA
power               : 1.00mW
```

SEE ALSO

get_user_units(2)
set_user_units(2)

report_unloaded_registers

Reports registers that were removed because their values were not used.

SYNTAX

string **report_unloaded_registers**

DESCRIPTION

Displays a list of registers that were removed from the design by the **compile** command because their values were unused. Unloaded register removal is controlled by the **compile.seqmap.remove_unread_registers** application option.

EXAMPLES

The following example shows a summary report:

```
shell> report_unloaded_registers
****************************************
Report : unloaded_registers
 ...

****************************************

Register
--------------------------------------------------------------------------------
bot/z0_reg
bot/z1_reg
bot/z2_reg
bot/c1_reg
1
```

SEE ALSO

compile_fusion(2)
compile.seqmap.remove_unloaded_registers(3)

report_user_units

Reports the currently specified input and output units.

SYNTAX

string **report_user_units**
  [-nosplit]

ARGUMENTS

-nosplit

Retains long lines in the output, even if the lines are longer than 80 columns. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing scripts to extract information from the report output.

DESCRIPTION

The **report_user_units** command displays the currently specified input and output units.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example reports the user units for the current design.

```
prompt> report_user_units

****************************************
Report : user_units
 ...

****************************************
```

```
Input Units (set to default)
-------------------------------------------------
time                : 1.00ns
resistance          : 1.00kOhm
capacitance         : 1.00pF
voltage             : 1.00V
current             : 1.00mA
power               : 1.00mW

Output Units (set to default)
-------------------------------------------------
time                : 1.00ns
resistance          : 1.00kOhm
capacitance         : 1.00pF
voltage             : 1.00V
current             : 1.00mA
power               : 1.00mW
```

SEE ALSO

get_user_units(2)
set_user_units(2)

report_utilization

Report utilization of the current block.

SYNTAX

float **report_utilization**
[-of_objects *objects*]
[-config *config*]
[-scope *config_scope*]
[-region *region_spec*]
[-grid_size *distance*]
[-verbose]

Data Types

objects	collection
config	string
distance	float
region_spec	list

ARGUMENTS

-of_objects *objects*

A homogeneous collection consisting of objects for which the utilization needs to be reported. In this case, each object in the list is a name, pattern or collection. The supported objects are

- **block**

  The block on which utilization is to be reported. Only one block is allowed with this option. If multiple blocks are specified, then utilization is reported only for the first block.

- **voltage_area**

  The utilization for the voltage area will be reported. If multiple voltage areas are specified, then the collective utilization of all voltage areas will be reported.

- **site_array**

  The utilization for the site-array will be reported. If multiple site arrays are specified, then the collective utilization of all site arrays will be reported.

- **bound**

  The utilization for the move-bound will be reported. If multiple move bounds are specified, then the collective utilization of all move bounds will be reported. Only move bounds are supported with this option.

-config *config*

Name (or collection) of the utilization configuration, created using *create_utilization_configuration*, to be used for reporting. If the input is a collection, then the first configuration in the collection will be used for reporting. In the absence of this option user defined default configuration would be used if any exists.

-scope *scope*

The values it can take are 'lib', 'tech' and 'block'. Restricts the search for a configuration to a particular scope. By default, the order of search is block, lib and then tech i.e. the configuration is first searched in the current block, then the current library and finally the tech associated with the current library.

-region

Specifies the region for which to report utilization. Specify a rectangle, polygon, or collection of geometric objects.

To specify a rectangle, use the following syntax to specify the lower-left and upper-right corners of the rectangle:

{{*llx lly*} {*urx ury*}}

To specify a polygon, use the following syntax to specify the coordinates of the polygon:

{{*x y*} {*x y*} {*x y*} {*x y*}...}

By default, the utilization of the current block is reported.

This option is mutually exclusive with the **-of_objects** option.

-grid_size *distance*

Specifies the grid size of each subregion. In addition to the entire block or a user-specified region, the command divides the given area (either entire chip or a user specified region if you specify the -region option) into a set of sub-regions and reports the corresponding utilization of the subregion to provide a better understanding of the evenness of the cell distribution. The grid_size is the size of each subregion. The minimum grid size is 3 times the height of the placement site_def. The unit for grid size is in microns. This option is mutually exclusive with the **-of_objects** option.

-verbose

Prints additional information in the report for utilization based on block boundary and for utilization based on site-rows of the block.

DESCRIPTION

The utilization calculation involves the computation of available area (Capacity) and the utilized area (Demand), and is calculated as :

Utilization = Demand / Capacity,

where,

Demand: Sum total of area occupied by the objects

Capacity: Total available area.

This command can be used to report the utilization of the current block. By default, with no options specified, the command reports utilization of the current block with respect to site-rows, and excluding the following:

- hard-macros

- soft-macros
- macro-keepouts
- io cells
- hard blockages

The utilization configurations created using the command "*create_utilization_configuration*" can be used to control the way utilization is calculated.

Capacity calculation:

For the utilization of a block, if a configuration is specified, then the capacity area calculation depends upon the value of "-capacity" option in the configuration.

For the utilization of other objects specified through '-of_objects', the capacity area calculation is essentially the effective boundary of the object. For voltage-area the total capacity is the effective region occupied by the constituent voltage-area shapes. For move-bounds the total capacity is the effective region occupied by the constituent move-bound regions. For site-arrays, the total capacity is the effecitve shape of the site-array based on the stacking order.

If the utilization configuration has the '-exclude' option set, then the effective regions of the objects corresponding to the exclusion types are deducted from the capacity region.

Demand calculation:

The default demand is the sum of the areas of all the cells in the block. All the cells, belonging to the container (block, voltage-area, move-bound or site-array) will be considered for demand calculation regardless of whether or not they overlap with the capacity region.

If two cells, to be considered as demand, overlap and either of them are not fixed then the demand of these two cells is the sum of areas of the individual cells. If they are fixed, then the demand will be the union of individual areas.

The hierarchical physical blocks will be handled as follows:

- If the physical block is of view type DESIGN then the demand
computation will be hierarchical and will follow the same rules for the
physical block contents as the top block

- If the physical block is of view type FRAME or OUTLINE
then the demand computation will not be hierarchical and the bounding box
of the physical block will be considered as demand

For the utilization of a block, with respect to its boundary, the '-verbose' option will report information about the number of cells outside or partially overlapping with the block-boundary and their aggregate area. For the utilization of a block, with respect to its site-rows, the '-verbose' option will report information about the independent utilization of all the site-definitions.

On success, the command returns as a string, the computed utilization upto 4 significant digits.

EXAMPLES

The following example reports utilization of the current block using the default configuration.

```
prompt> report_utilization
****************************************
Report : report_utilization
Design : TEST1
Version: I-2014.03
Date   : Mon Mar 10 00:39:48 2014
```

```
****************************************
Utilization Ratio:                0.6174
Utilization options:
- Area calculation based on:         site_row of block TEST1
- Categories of objects excluded:     hard_macros macro_keepouts soft_macros
                                io_cells hard_blockages
Total Area:                    1967889.5460
Total Capacity Area:            442954.4265
Total Area of cells:            273477.8556
Area of excluded objects:
 - hard_macros    :             573858.0990
 - macro_keepouts  :              0.0000
 - soft_macros    :             930543.9417
 - io_cells       :            1492840.8000
 - hard_blockages  :             121431.1025

0.6174
```

The following example reports utilization of the current block with respect to its boundary and no-exclusions.

```
prompt> create_utilization_configuration config1 -capacity boundary
prompt> report_utilization -config config1
****************************************
Report : report_utilization
Design : TEST1
Version: I-2014.03
Date   : Mon Mar 10 00:46:03 2014
****************************************
Utilization Ratio:                0.6744
Utilization options:
 - Area calculation based on:         boundary of block TEST1
 - Categories of objects excluded:     None
Total Area:                    4639478.6848
Total Capacity Area:            4639478.6848
Total Area of cells:            3128643.2023

0.6744
```

The following example reports utilization of the current block with respect to its boundary and excluding the fixed_cells.

```
prompt> create_utilization_configuration config1 \
  -capacity boundary -exclude {fixed_cells}
prompt> report_utilization -config config1 -verbose
****************************************
Report : report_utilization
Design : TEST1
Version: I-2014.03
Date   : Mon Mar 10 04:14:52 2014
****************************************
Utilization Ratio:                0.6298
Utilization options:
 - Area calculation based on:         boundary of block TEST1
 - Categories of objects excluded:     fixed_cells
Total Area:                    4639478.6848
Total Capacity Area:            4081512.9264
Total Area of cells:            2570677.4439
Area of excluded objects:
```

```
 - fixed_cells    :              557965.7584

  Cells partially overlapping with block boundary:
  Total number of cells:                  72
  Aggregate area of non-overlapping region:    2687.0400

  0.6298
```

The following example reports utilization of the current block with respect to the site-rows and excluding hard-macros and macro-keepouts

```
  prompt> create_utilization_configuration config_sr \
    -capacity site_row -exclude {hard_macros macro_keepouts}
  prompt> report_utilization -config config_sr -verbose
  ****************************************
  Report : report_utilization
  Design : TEST1
  Version: I-2014.03
  Date   : Mon Mar 10 04:18:39 2014
  ****************************************
  Utilization Ratio:              0.8637
  Utilization options:
   - Area calculation based on:       site_row of block TEST1
   - Categories of objects excluded:    hard_macros macro_keepouts
  Total Area:                  1967889.5460
  Total Capacity Area:            1394031.4470
  Total Area of cells:            1204021.7973
  Area of excluded objects:
   - hard_macros    :            573858.0990
   - macro_keepouts  :            64220638.2300

  Utilization of site-rows with:
   - Site 'unit':               86.37%

  0.8637
```

The following example reports utilization of the voltage-area DEFAULT_VA

```
  prompt> report_utilization -of_objects [get_voltage_areas]
  ****************************************
  Report : report_utilization
  Design : test
  Version: I-2013.12
  Date   : Thu Nov 21 02:43:16 2013
  ****************************************
  Utilization Ratio:              0.6095
  Utilization options:
   - Area calculation based on:       Voltage area DEFAULT_VA
   - Categories of objects excluded:    None
  Total Area:                  108372.4800
  Total Capacity Area:            108372.4800
  Total Area of cells:            66054.4000

  0.6095
```

The following example reports utilization of the move-bound MB1

```
  prompt> report_utilization -of_objects [get_bounds MB1]
```

```
****************************************
Report : report_utilization
Design : test
Version: I-2013.12
Date   : Thu Nov 21 02:43:29 2013
****************************************
Utilization Ratio:                0.1708
Utilization options:
 - Area calculation based on:         Move bound MB1
 - Categories of objects excluded:    None
Total Area:                    9519.3600
Total Capacity Area:           9519.3600
Total Area of cells:           1625.6000

0.1708
```

SEE ALSO

create_utilization_configuration(2)
get_utilization_configurations(2)
remove_utilization_configurations(2)

report_vclp_options

This command is used to report the VCLP related option settings.

SYNTAX

status **report_vclp_options**

DESCRIPTION

The **report_vclp_options** command is used to report the VCLP related option settings. This command can be used to check if the settings are correct or need to be changed.

SEE ALSO

check_vclp_design(2)

report_versions

Displays information about release versions supported for saving and committing libraries.

SYNTAX

status **report_versions**

DESCRIPTION

The **report_versions** command displays information about the release versions supported by save_lib -as -version in an implementation tool and commit_workspace -version in the library manager.

Compatibility in the report means the tools listed are natively on this schema revision. They can read all the data in the current schema, and create the database in the same schema too.

Forward compatibility means the tools listed can read the database in newer schema revision in a way that, any object which is specific to the newer schema revision is ignored, while all the already supported objects are read by the tools. Please refer to **Includes support for** section in the same report for details of the new schema objects attached to the specific schema revision.

EXAMPLE

The following command reports the currently supported versions:

```
prompt> report_versions

J-2014.06
  ICC2 schema revision: 1.0
  Compatibility:
     - ICV J-2014.06-SP1

J-2014.06-SP1
  ICC2 schema revision: 1.001
  Includes Support for:
     - Analog Constraints
     - Dynamic Power
     - finFet Grid
     - net min/max routing layers
     - viaDef attributes

1
```

SEE ALSO

save_lib(2)
commit_workspace(2)

report_via_defs

Generates a report of via_defs in a specified block or tech.

SYNTAX

string **report_via_defs**
[-design *design*]
[-library *library*]
[-tech *tech*]
[-verbose]
[-significant_digits *digits*]
[-nosplit]
[*via_def_list*]

Data Types

design collection
library collection
tech collection
digits integer
via_def_list collection

ARGUMENTS

-design *design*

Specifies the block for finding via_defs. If neither a design nor tech is specified, the current block is used.

-library *library*

Specifies the tech for finding via_defs. The tech contained by or referenced by the given library is used. If neither a design nor tech is specified, the current block is used.

-tech *tech*

Specifies the tech for finding via_defs. If neither a design nor tech is specified, the current block is used.

-verbose

Specifies whether the report contains additional columns of data, reporting more complete information about the given via_defs.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default value is 2. Set this application option to override the default.

-nosplit

Most of the via_def information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing software to extract information from the report output.

via_def_list

Specifies a list of via_defs in the given block or tech to be reported. The default is to report all via_defs in the given container.

DESCRIPTION

The command displays information about the via_defs in the block or tech specified. Attributes such as the cut and metal layers of the via_def are displayed.

EXAMPLES

The following example reports all via_defs in the current block.

```
prompt> report_via_defs

****************************************
Report : report_via_defs
Design : top
 ...

****************************************
Via_def              Type    Lower  Cuts   Upper  Is_default
--------------------------------------------------------------------------
FATVIA12_1_2          custom  M1     VIA1   M2     --
FATVIA12_2_1          custom  M1     VIA1   M2     --
VIA12_r90             custom  M1     VIA1   M2     --
2x_spacing_clocks_VIA12  custom  M1     VIA1   M2     --
VIA23_r90             custom  M2     VIA2   M3     --
2x_spacing_clocks_VIA23  custom  M2     VIA2   M3     --
2x_spacing_clocks_VIA34  custom  M3     VIA3   M4     --
2x_spacing_clocks_VIA45  custom  M4     VIA4   M5     --
2x_spacing_clocks_VIA56  custom  M5     VIA5   M6     --
2x_spacing_clocks_VIA67  custom  M6     VIA6   M7     --
```

The following example displays a verbose via_def report for library 'test'.

```
prompt> report_via_defs -verbose -library test

****************************************
Report : report_via_defs
Tech   : ABC N90
 ...

****************************************
```

```
Via_def    Type    Lower  Cuts   Upper  Is_default
  Source_type       Cut_size   Lower_enc Upper_enc
    Min_rows Min_cols Min_cut_spacing  Is_excluded_for_signal_route
--------------------------------------------------------------------------------
CONT1      simple  PO     CO     M1     default
  single_cut_fixed    0.12 0.12   0.05 0.05  0.05 0.00
    --       --       --           --
VIA12      simple  M1     VIA1   M2     default
  single_cut_fixed    0.13 0.13   0.01 0.05  0.01 0.05
    --       --       --           --
VIA12_OPTI simple  M1     VIA1   M2     --
  single_cut_fixed    0.13 0.13   0.01 0.08  0.01 0.08
    --       --       --           --
FATVIA12   simple  M1     VIA1   M2     --
  single_cut_fixed    0.13 0.13   0.05 0.05  0.05 0.05
    --       --       --           --
VIA23      simple  M2     VIA2   M3     default
  single_cut_fixed    0.13 0.13   0.01 0.05  0.05 0.01
    --       --       --           --
VIA23_OPTI simple  M2     VIA2   M3     --
  single_cut_fixed    0.13 0.13   0.01 0.08  0.08 0.01
    --       --       --           --
FATVIA23   simple  M2     VIA2   M3     --
  single_cut_fixed    0.13 0.13   0.05 0.05  0.05 0.05
    --       --       --           --
FATVIA34   simple  M3     VIA3   M4     --
  single_cut_fixed    0.13 0.13   0.05 0.05  0.05 0.05
    --       --       --           --
```

SEE ALSO

get_via_defs(2)
shell.common.report_default_significant_digits(3)

report_via_ladder_candidates

Reports via ladder candidates set by the **set_via_ladder_candidate** command.

SYNTAX

status **report_via_ladder_candidates**
list_of_lib_pins

Data Types

list_of_lib_pins pin names or collection

ARGUMENTS

list_of_lib_pins

Specifies list of library-pins for which via ladder candidates will be reported.

DESCRIPTION

This command reports existing via ladder candidates on the specified library-pins. This command returns 1 if succeeded, 0 otherwise.

EXAMPLES

The following example reports all via ladder candidates for the library-pin.

```
prompt> report_via_ladder_candidates [get_lib_pin tcbn90ghvt/AN2HVTD0/A1]
Lib Pin              Via Ladder Candidates
--------------------------------------------------------------------------------
tcbn90ghvt/AN2HVTD0/A1  VL1 VL2 VL3
1
```

SEE ALSO

set_via_ladder_candidate(2)
reset_via_ladder_candidates(2)
set_via_ladder_constraints(2)
remove_via_ladder_constraints(2)
report_via_ladder_constraints(2)

report_via_ladder_constraints

Reports via ladder constraints set by the **set_via_ladder_constraints** command.

SYNTAX

status **report_via_ladder_constraints**
[-pins *pins*]
[-all]
[-nosplit]

Data Types

pins pin names or collection

ARGUMENTS

-pins *pins*

Specifies pins for which via ladder constraints will be reported. Pins must have block context and should belong to current block.

-all

Enables reporting of consolidated list of via ladders in prioritized order for the pins specified using the **-pins** option. The consolidated list comprises the following via ladders in descending order of priority:

- **the tool specified via ladder constraint**: via ladders auto inferred by the tool.
- **the user specified via ladder constraint**: via ladders set using **set_via_ladder_constraints**.
- **the user specified via ladder rules**: via ladders set using **set_via_ladder_rules**.
- **electro-migration via ladder candidates**: subset of via ladders set using **set_via_ladder_candidate**. Only the electro-migration type via ladders are union merged to the consolidated list if the library-pin has the **is_em_via_ladder_required** attribute specified as true. If any via ladder has both **for_electro_migration** and **for_high_performance** attribute set to true it would appear after all **for_electro_migration** via ladders that has **for_high_performance** attribute set to false.
- **pattern must join via ladder**: a representative via ladder with generated name is appended to the consolidated list if the library-pin has the **pattern_must_join** attribute specified as true. The generated via ladder name has the format **VL_auto_<n>_*** where n is the count of top-layer pin shapes for the pattern must join port.

  If you specify this option, you must also specify the **-pins** option.

-nosplit

Does not split lines if column overflows.

DESCRIPTION

This command reports existing via ladder constraints. If pins are specified with "-pins" option, then constraints are reported for only those pins. Otherwise, all existing constraints are reported. For every pin specified with "-pins" option, for which via ladder constraints were not set, warning is printed.

This command returns 1 if succeeded, 0 otherwise.

EXAMPLES

The following example reports all via ladder constraints from the block.

```
prompt> report_via_ladder_constraints
Pin             Via ladders
--------------------------------------------------------------------------------
u1/i1           VL1,VL3
u1/i2           VL4,VL6,VL8
1
```

The following example reports via ladder constraints for specified pins only.

```
prompt> report_via_ladder_constraints -pins "u1/i1"
Pin             Via ladders
--------------------------------------------------------------------------------
u1/i1           VL1,VL3
1
```

The following example reports consolidated list of via ladder for specified pin that has both **is_em_via_ladder_required** and **pattern_must_join** attributes set as true on its library-pin. In this example VL_EM_1, VL_EM_2 and VL_EM_PERF_1 are all electro-migration type via ladder candidates and VL_EM_PERF_1 is also the only performance type via ladder candidate. The generated name of pattern must join via ladder is VL_auto_2_* where 2 is the top-layer pin shape count of the pattern must join port.

```
prompt> report_via_ladder_constraints -all -pins "u1/i1"
Pin             Via ladders
--------------------------------------------------------------------------------
u1/i1           VL1,VL3,VL_EM_1,VL_EM_2,VL_EM_PERF_1,VL_auto_2_*
1
```

SEE ALSO

set_via_ladder_constraints(2)
remove_via_ladder_constraints(2)
set_via_ladder_candidate(2)
reset_via_ladder_candidates(2)
report_via_ladder_candidates(2)

report_via_ladder_rules

Reports via ladder rules set by the **set_via_ladder_rules** command.

SYNTAX

status **report_via_ladder_rules**
[-nosplit]

ARGUMENTS

-nosplit

Does not split lines if column overflows.

DESCRIPTION

This command reports existing via ladder rules.

This command returns 1 if succeeded, 0 otherwise.

EXAMPLES

The following example reports all via ladder constraints from the block.

```
prompt> report_via_ladder_rules
****************************************
Report : report_via_ladder_rules
Design : r4000
Version: K-2015.06-SP3-BETA
Date   : Mon Sep  7 04:48:12 2015
****************************************
-all_instances_of :          mAlu/PC[30],mRegister/Clk
-default_ladders :           VIA12,VIA56_OPTI
-all_clock_outputs :          true
-all_clock_inputs :          false
-all_pins_driving :          Clk
-master_pin_map :
```

```
cell_master/pin                 via ladders list
-------------------------------------------------------------------------------
mAlu/PC[26]                     VIA56_OPTI
mAlu/PC[31]                     CONT1
1
```

SEE ALSO

set_via_ladder_rules(2)
remove_via_ladder_rules(2)

report_via_ladders

Generates a report of via ladders in the current design.

SYNTAX

string **report_via_ladders**
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*via_ladder_list*]

Data Types

digits integer
via_ladder_list collection

ARGUMENTS

-verbose

Specifies whether the report contains additional columns of data, reporting more complete information about the given via ladders.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this application option if you want to override the default. Fixed-precision floating-point arithmetics is used for delay calculation; therefore, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h. The upper bound on a meaningful value of the argument to **-significant_digits** is then FLT_DIG - ceil(log10(fabs(*any_reported_value*))).

-nosplit

Most of the via ladder information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing software to extract information from the report output.

via_ladder_list

Specifies a list of via ladders in the given block to be reported. The default is to report all via ladders in the given container.

DESCRIPTION

The command displays information about the via ladders in the block specified. Attributes such as the via_rule_name and name of the via ladder are displayed.

EXAMPLES

The following example reports a via_ladder named "VIA_LADDER_0" in the current design.

prompt> **report_via_ladders VIA_LADDER_0**

```
Via_ladder    Via_rule     EM      PMJ     HP
-------------------------------------------------------------------------------
VIA_LADDER_0  rule_1       false   true    false
```

SEE ALSO

create_via_ladder(2)
insert_via_ladders(2)
get_via_ladders(2)
remove_via_ladders(2)
shell.common.report_default_significant_digits(3)

report_via_mapping

Report the via-mapping options used for redundant via insertion.

SYNTAX

status **report_via_mapping**
[-nosplit]
[-from {*via_pattern*}]
[-to {*via_pattern*}]

Data Types

via_pattern string

ARGUMENTS

-nosplit

Does not split lines if column overflows.

-from {*via_pattern*}

Specifies the via-pattern corresponding to the vias to be replaced during redundant via insertion.

Any via-mapping whose "to be replaced" via matches with this pattern will be reported.

A via-pattern has two parts: viaDefName and site_spec mxn. viaDefName is via definition name. mxn specifies the number of contacts in the horizontal (m) and vertical (n) directions.

-to {*via_pattern*}

Specifies the via-pattern corresponding to the vias to be inserted during redundant via insertion.

Any via-mapping option whose "replacement" via matches with this pattern will be reported.

DESCRIPTION

The **report_via_mapping** command reports the existing via-mapping options from the current block. By default, all the via-mappings defined in the current block will be reported.

EXAMPLES

The following example reports the via-mapping options for the replacement of via with via-def 'VIA12':

```
prompt> report_via_mapping  -from {VIA12 1x1}
```

The following example reports the via-mapping options where, the replacement via is a double via with via-def 'VIA12':

```
prompt> report_via_mapping -to {VIA12 1x2}
```

The following example reports the via-mapping options for via replacement from vias with via-def 'VIA12' to a double via with via-def 'VIA12':

```
prompt> report_via_mapping -from {VIA12 1x1} -to {VIA12 1x2}
```

SEE ALSO

remove_via_mappings(2)
add_via_mapping(2)

report_via_matrixes

Generates a report of via_matrixes in the current design.

SYNTAX

string **report_via_matrixes**
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*via_matrix_list*]

Data Types

digits integer
via_matrix_list collection

ARGUMENTS

-verbose

Specifies whether the report contains additional columns of data, reporting more complete information about the given via_matrixes.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this application option if you want to override the default. Fixed-precision floating-point arithmetics is used for delay calculation; therefore, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h. The upper bound on a meaningful value of the argument to **-significant_digits** is then FLT_DIG - ceil(log10(fabs(*any_reported_value*))).

-nosplit

Most of the via_matrix information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing software to extract information from the report output.

via_matrix_list

Specifies a list of via_matrixes in the given block or tech to be reported. The default is to report all via_matrixes in the given container.

DESCRIPTION

The command displays information about the via_matrixes in the block or tech specified. Attributes such as the via_def and name of the via_matrix are displayed.

Note: The results of 'Lower Mask Constraint', 'Cut Mask Constraint' and 'Top Mask Constraint' are only reported for 1000 base vias.

EXAMPLES

The following example reports a via matrix named VIA_MATRIX_0 in the current block.

```
prompt> report_via_matrixes VIA_MATRIX_0

Via_matrix    Via_def   Origin
--------------------------------------------------------------------------------
VIA_MATRIX_0  VIA12     {1000 200}
```

The following example displays verbose information for the via matrix named VIA_MATRIX_0.

```
prompt> report_via_matrix -verbose VIA_MATRIX_0
Via_matrix    Via_def   Origin          Description
--------------------------------------------------------------------------------
VIA_MATRIX_0  VIA12     {1000 200}      Orientation:  R90
                                        Base Pitch:   {10 10}
                                        Base Size:    {20 30}
                                        Pitch:        {10 10}
                                        Size:         {20 30}
                                        Shape Use:    detail_route
                                        Net:          Clk
                                        Lower Mask Pattern:
                                                arbitrary
                                        Lower Mask Constraint:
                                                {{0 0 no_mask}
                                                 {0 1 mask_one}
                                                 {1 1 mask_two}}
                                        Cut Mask Pattern:
                                                uniform
                                        Cut Mask Constraint:
                                                {{0 0 no_mask}}
                                        Top Mask Pattern:
                                                arbitrary
                                        Top Mask Constraint:
                                                {{0 0 no_mask}
                                                 {0 1 mask_one}
                                                 {1 1 mask_two}}
                                        Base Pattern: 10 01
                                        Via Pattern   1111
```

SEE ALSO

create_via_matrix(2)
get_via_matrixes(2)
remove_via_matrixes(2)
report_via_matrixes(2)
shell.common.report_default_significant_digits(3)

report_via_regions

Generates a report of via_regions in a specified frame block.

SYNTAX

string **report_via_regions**
[-design *design*]
[-verbose]
[-significant_digits *digits*]
[-nosplit]
[*via_region_list*]

Data Types

design collection
digits integer
via_region_list collection

ARGUMENTS

-design *design*

Specifies the block for finding via_regions. If no design is specified, the current block is used.

-verbose

Specifies whether the report contains additional columns of data, reporting more complete information about the given via_regions.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default value is 2. Use this application option to override the default.

-nosplit

Most of the via_def information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The **-nosplit** option prevents line-splitting and facilitates writing software to extract information from the report output.

via_region_list

Specifies a list of via_regions in the given block to be reported. The default is to report all via_regions in the given container.

DESCRIPTION

The command displays information about the via_regions in the block specified. Attributes such as the owner, via_def and type are displayed.

EXAMPLES

The following example reports all via_regions in the current block.

```
prompt> report_via_regions

****************************************
Report : Report via regions
 ...

****************************************
ViaRegion       Owner      ViaDef    Type    Rect Rotate
--------------------------------------------------------------------------------
Reset/VR_0      Reset      VIA12     User    Yes  --

MemData[31]/VR_0  MemData[31] CONT1     User    Yes  --

MemData[31]/VR_1  MemData[31] FATVIA23  User    Yes  Yes

MemData[31]/VR_2  MemData[31] CONT1     User    Yes  --
```

The following example displays a verbose via_region report.

```
prompt> report_via_regions -verbose
report_via_regions -verbose
****************************************
Report : Report via regions
 ...

****************************************
ViaRegion       Owner      ViaDef    Type    Rect Rotate  BBox
-------------------------------------------------------------------------------
Reset/VR_0      Reset      VIA12     User    Yes  --     {104.9 133.42} {105 133.56}

MemData[31]/VR_0  MemData[31] CONT1     User    Yes  --     {113.53 133.42} {113.67 133.56}

MemData[31]/VR_1  MemData[31] FATVIA23  User    Yes  Yes    {113.53 133.42} {113.6 133.5}

MemData[31]/VR_2  MemData[31] CONT1     User    Yes  --     {113.53 133.42} {113.6 133.5}
```

SEE ALSO

get_via_regions(2)
create_via_region(2)
shell.common.report_default_significant_digits(3)

report_via_rules

Generates a report of via rules in a specified block or tech.

SYNTAX

string **report_via_rules**
[-design *design*]
[-library *library*]
[-tech *tech*]
[-verbose]
[-nosplit]
[-all]
[*via_rule_list*]

Data Types

design collection *library* collection *tech* collection *via_rule_list* string or collection

ARGUMENTS

-design *design*

Specifies the block for finding via rules. If neither a design nor tech is specified, the current block is used. If current block is not set then current lib is used.

-library *library*

Specifies the library for finding via rules. The tech contained by or referenced by the given library is used. If neither a design nor tech is specified, the current block is used. If current block is not set then current lib is used.

-tech *tech*

Specifies the tech for finding via rules. If neither a design nor tech is specified, the current block is used. If current block is not set then current lib is used.

-verbose

Outputs the various set properties names and their values for each of the via rules along with default information. Name of the properties displayed is same as used in the technology file.

-nosplit

Prevents line splitting if the column overflows. Most information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

-all

Displays all via rules in the specified scope. You must specify either **-all** or *via_rule_list*.

via_rule_list

Displays information only on the specified objects in *via_rule_list*. The argument contains either a collection of via rules or a pattern that matches the via_rule names in the specified scope. You must specify either **-all** or *via_rule_list*.

DESCRIPTION

This command displays information about via rules in the block or technology file. By default, the command displays the via rules in both block and tech scope. If the search is to be limited to either block or tech scope, the same can be specified as an option. By default, the command produces a brief report containing via_rule's name and if they are a via ladder. You must specify either **-all** or *via_rule_list*.

EXAMPLES

The following example outputs the report for all via rule objects.

```
prompt> report_via_rules -all
****************************************
Report : report_via_rules
Version:
Date   :
****************************************
Name             Via ladder
------------------------------------------
VR1              No
VL1              Yes
1
```

The following example outputs the verbose report for the via rule named VL1.

```
prompt> report_via_rules VL1 -verbose
****************************************
Report : report_via_rules
Version: L-2016.03-SP5-1-BETA
Date   : Sun Oct 23 23:27:33 2016
****************************************
Name   Via ladder Property name        Value
-----------------------------------------------------
VL1    Yes       cutLayerNameTblSize   2
                 cutLayerNameTbl       VIA1, VIA2
                 cutNameTbl            cut1, cut2
                 numCutRowsTbl         2, 2
                 numCutsPerRowTbl      2, 2
1
```

SEE ALSO

create_via_rule(2)
get_via_rules(2)
remove_via_rules(2)
report_cells(2)
report_hierarchy(2)
report_nets(2)
report_references(2)

report_virtual_pads

Reports existing virtual pads.

SYNTAX

status **report_virtual_pads**

ARGUMENTS

This command has no arguments

DESCRIPTION

This command reports all existing virtual pads.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example reports all existing virtual pads.

prompt> **report_virtual_pads**

SEE ALSO

analyze_power_plan(2)
read_virtual_pad_file(2)
remove_virtual_pads(2)
set_virtual_pad(2)
write_virtual_pad_file(2)

report_voltage_area_rules

Reports voltage area rules in the design.

SYNTAX

status **report_voltage_area_rules**
  [-nosplit]
  [*voltage_area_rules*]

Data Types

voltage_area_rules collection

ARGUMENTS

-nosplit

Does not split lines if column overflows.

voltage_area_rules

Specifies the voltage area rules to report. This can be a collection handle of voltage area rules or names of patterns. You can use the **get_voltage_area_rules** command to specify objects. If not specified, the tool reports all voltage area rules in the design.

DESCRIPTION

This command reports the details of the specified voltage area rules. Details include the rule name, whether or not pass through and physical feedthrough are allowed, and a list of voltage areas constrained by the rule.

EXAMPLES

The following example reports the voltage area rule named "RULE1" in the design.

prompt> **report_voltage_area_rules RULE1**

The following example reports all voltage area rules in the design.

prompt> **report_voltage_area_rules**

SEE ALSO

create_voltage_area_rule(2)
get_voltage_area_rules(2)
remove_voltage_area_rules(2)

report_voltage_areas

Reports the voltage areas in the design.

SYNTAX

status **report_voltage_areas**
[-design design]
[-hierarchical]
[-verbose]
[-nosplit]
[-significant_digits *digits*]
[*voltage_area_list*]

Data Types

digits int
voltage_area_list list

ARGUMENTS

-design *design*

Specifies the top design for finding objects. If this is not specified, objects will be found in the current design.

-hierarchical

Show hierrachical voltage_area info.

-verbose

Displays verbose voltage area information.

-nosplit

Does not split lines if column overflows.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the **shell.common.report_default_significant_digits** application option, whose default is 2. Use this option if you want to override the default.

voltage_area_list

Specifies a list of voltage areas to report. The list may contain voltage area names, patterns, or collections. A collection may be specified by using the **get_voltage_areas** command. This argument is mutually exclusive with the **-all** option. You must specify

one, but not both.

DESCRIPTION

This command reports the user-specified voltage area constraints in the design, as specified by the **create_voltage_area** command. If you use the **-all** option, all voltage areas in the design, created by the **create_voltage_area** command are reported.

The report includes voltage area name, target utilization rate, the power domain associated with this voltage area, the voltage area shapes, guardbands, the voltage boundary coordinates, and the relative stacking order.

EXAMPLES

The following example reports the voltage area *cntrl/d1*:

```
prompt> report_voltage_areas cntrl/d1 -verbose

Voltage_area       Description
-------------------------------------------------------------------------------
cntrl/d1          not_fixed with 5 shapes target_utilization 0.5
                  power_domain: cntrl/d1
                  VOLTAGE_AREA_SHAPE_1
                  guard_band:       (1.00, 2.00)
                  boundary:         (0.00, 0.00)
                                    (0.00, 60.00)
                                    (60.00, 60.00)
                                    (60.00, 0.00)
                  stacking_order:   1

                  VOLTAGE_AREA_SHAPE_2
                  guard_band:       (1.00, 1.00)
                  boundary:         (0.00, 0.00)
                                    (0.00, 70.00)
                                    (70.00, 70.00)
                                    (70.00, 0.00)
                  stacking_order:   2

                  VOLTAGE_AREA_SHAPE_3
                  guard_band:       (2.00, 2.00)
                  boundary:         (0.00, 0.00)
                                    (0.00, 80.00)
                                    (80.00, 80.00)
                                    (80.00, 0.00)
                  stacking_order:   3

                  VOLTAGE_AREA_SHAPE_4
                  guard_band:       (3.00, 3.00)
                  boundary:         (0.00, 0.00)
                                    (0.00, 90.00)
                                    (90.00, 90.00)
                                    (90.00, 0.00)
```

```
        stacking_order:   4

        VOLTAGE_AREA_SHAPE_5
        guard_band:      (4.00, 4.00)
        boundary:        (0.00, 0.00)
                         (0.00, 95.00)
                         (95.00, 95.00)
                         (95.00, 0.00)
        stacking_order:   5

1
```

SEE ALSO

create_voltage_area(2)
remove_voltage_areas(2)
get_voltage_areas(2)
create_voltage_area_shape(2)
remove_voltage_area_shapes(2)
get_voltage_area_shapes(2)

report_wrapper_configuration

Reports the wrapper configuration of the current design.

SYNTAX

status **report_wrapper_configuration**

ARGUMENTS

The **report_wrapper_configuration** command has no arguments.

DESCRIPTION

This command reports the wrapper configuration applied by the **set_wrapper_configuration** command.

EXAMPLES

The following example shows a wrapper configuration report.

```
prompt> set_wrapper_configuration -reuse_threshold 5
1
prompt> report_wrapper_configuration
****************************************
Report : Wrapper Specification
Design : top
Version: P-2019.03
Date   : Thu Feb  7 08:03:48 2019
****************************************

----------------------------------------
TEST MODE: all_dft
VIEW    : Specification
----------------------------------------
Wrapper Configuration
=====================
Reuse Threshold:                5
```

```
Hier Wrapping:                    Enable
Add Wrapper Cells To Power Domain:    Enable
Mix Cells:                        False
Mix With Scan Cells:              True
Use System Clock For Dedicated Wrapper Cells: Enable
Depth Threshold:                  Unspecified
Chain count:                      Unspecified
Gate Cells:                       none
```

SEE ALSO

set_wrapper_configuration(2)

reset_app_options

Resets application options to the unset status on the specified block.

SYNTAX

integer **reset_app_options**
[-block *block*]
[-user_default]
[-unknown]
names

OLD SYNTAX

integer **reset_app_options**
[-design *design*]
[-global]
names

Data Types

block block name or collection
names list

ARGUMENTS

-block *block*

The name of the block on which the option values will be reset. This is an optional option and cannot be specified with the *-user_default* option. If this option is not specified and the application option being reset it design scoped, then the value on the current block with be reset.

-user_default

Indicates that the user-default for the specified option will be reset. This option cannot be specified with the *-block* option.

-unknown

Indicates that the value for the unknown option in the specified block will be reset. If the block is not specified, the value of the current block's unknown app option will be reset.

names

Specifies a list of option names. An option name can include the wildcard characters asterisk (*) and question mark (?). "*" will match zero or more of any character, and "?" will match any single character. Note that if any of the names specified with no wild cards is invalid, this command will fail with no effect.

-design *design*

The name of the design on which the option values will be reset. This option cannot be specified with the *-global* option, and either *-design* or *-global* must be specified. This option is deprecated, use *-block* option instead.

-global

Indicates that the specified option values will be reset in the global scope. This option cannot be specified with the *-design* option. This option is deprecated.

DESCRIPTION

This command resets the values of the specified application options to unset status, on the specified block, or in the global scope, depending on the scope of the application option.

Each design will store only the options explicitly set on it. Any option values which are unset on the design will be obtained from the user-default or system-default (if user-default does not exist).

For application options that are of type "list of {name value} pairs", the subscript mechanism can be used to remove specific keys from the option value.

For backward compatibility the older syntax of the command will still be supported, but the options are made hidden. The option '-design', is made hidden.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the options.

EXAMPLES

The following example unsets the value of the *time.enable_preset_clear_arcs* option on the current block.

```
prompt> reset_app_options -block [current_block] {time.enable_preset_clear_arcs}
1
```

The following example unsets the value of the *shell.tmp_dir_path* option and the *time.enable_preset_clear_arcs* option, in the global scope.

```
prompt> reset_app_options {shell.tmp_dir_path time.enable_preset_clear_arcs}
1
```

The following example unsets the user default value of the *gui.enable_custom_setup_files*

```
prompt> reset_app_options -user_default {gui.enable_custom_setup_files}
1
```

The following example removes the key 'M4' from the current block's value of *test.layer_name_number_pair_list* option.

```
prompt> get_app_option_value -name test.layer_name_number_pair_list
```

```
M1 31 M2 32 M3 33 M4 34
prompt> reset_app_options -name test.layer_name_number_pair_list(M4)
1
prompt> get_app_option_value -name test.layer_name_number_pair_list
M1 31 M2 32 M3 33
```

SEE ALSO

set_app_options(2)
get_app_option_value(2)
get_app_options(2)
report_app_options(2)
help_app_options(2)

reset_cell_mode

Resets cell modes for cell instances.

SYNTAX

status **reset_cell_mode**
[*cell_list*]

Data Types

cell_list list

ARGUMENTS

cell_list

Specifies a list of instances for which the specified cell modes are to be made inactive. If not specified, reset cell modes for all cell instances in the design with cell mode settings.

DESCRIPTION

Removes the active mode for specified cell instances or for all instances in the design.

If this cell is a moded-ETM cell created with the library manager ETM flow, calling the **reset_cell_mode** command disables all modes of the cell, and all moded generated clocks and moded case values are removed.

If this is a cell with modes defined in the logic library source file, calling the **reset_cell_mode** command makes all modes of the cell active at once.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following command resets cell mode information for the RAM instance named Uram1.

prompt> **reset_cell_mode Uram1**

SEE ALSO

set_cell_mode(2)
report_cell_mode(2)

reset_checkpoints

Reset the logging of the checkpoint system and clears the stored state of previous checkpoints.

SYNTAX

integer **reset_checkpoints**
[-from *checkpoint_name*]

Data Types

checkpoint_name string

ARGUMENTS

-from *checkpoint_name*

Reset the checkpoint information from the gvien checkpoint name. This is an optional option.

DESCRIPTION

This command does either a full or partial reset of previously captured checkpoint history. By default, it resets all history, with the following effects:

- The ./checkpoint/checkpoint_history.rpt file will show no checkpoints
- The get_checkpoint_data -list_names command will show no checkpoints
- The ./checkpoint directory gets fully reset. If there is an existing ./checkpoint directory, it is backed up to a timestamped directory in the run directory called ./checkpoint_YYYY_MM_DD_hh:mm:ss, and a new ./checkpoint directory is created. Initially, this new checkpoint directory will have no contents except an unpopulated checkpoint_history.rpt file.

If the reset_checkpoints -from option is used, the checkpoint history is reset starting from the specified checkpoint. For example, if checkpoints A, B, C, D had been run, and reset_checkpoints -from C is called, then the history of only C and D are removed, but A and B remain:

- The ./checkpoint/checkpoint_history.rpt file will show history prior to the specified checkpoint, but none from that checkpoint
- The get_checkpoint_data -list_names command will show only the checkpoints prior to the -from checkpoint
- The get_checkpoint_data -name <checkpoint_name> command can be called only for those checkpoints prior to the -from checkpoint

- The ./checkpoint directory does not get fully reset. No backup is made. The only change to that directory is in the contents of the checkpoint_history.rpt file. Any other outputs in that directory generated by checkpoint reports will remain, even for those checkpoints that were reset. The user must manually clean-up those files if they wish to do so, as the checkpoint system has no knowledge of which files were written for which checkpoint.

Resetting the checkpoint system does not reset any report or action definitions or associations. All of those settings remain intact. Only the history of executed checkpoints is affected.

A typical application of using reset_checkpoints is when a user wants to restart a run through the flow scripts in the same directory where they previously ran. Or, when a user copies a previous run directory to a new directory to modify for a new experiment. If the new run does not pick up where the previous run left off, it is sensible to reset or partially reset the checkpoint history.

Resetting checkpoints is not the same as disabling the checkpoint system. Any checkpoint history is cleared, but the system is still active, and new eval_checkpoint commands encountered in the flow scripts can still trigger their associated actions and reports.

EXAMPLES

In this example, a user wishes to restart a run from the beginning of the flow in a previously existing run directory that used the checkpoint system. To avoid a duplicated checkpoint history, they start off by resetting the checkpoints:

```
prompt> reset_checkpoints
```

In this example, a user previously ran through place_opt, clock_opt, route, and route_opt flow steps, but later found problems in the setup requiring a rerun from the clock_opt step. Their first checkpoint name in that step was called clock_opt_route_clock, so they reset the checkpoints from that checkpoint name at the start of their rerun, to avoid duplicating the checkpoints from clock_opt onward:

```
prompt> reset_checkpoints -from clock_opt_route_clock
```

SEE ALSO

create_checkpoint_action(2)
remove_checkpoint_actions(2)
create_checkpoint_report(2)
remove_checkpoint_reports(2)
associate_checkpoint_action(2)
associate_checkpoint_report(2)
get_current_checkpoint(2)
set_checkpoint_options(2)
eval_checkpoint(2)
get_checkpoint_data(2)

reset_clock_gate_latency

Resets all clock-latency values previously applied to registers and clock-gating cells.

SYNTAX

status **reset_clock_gate_latency**

ARGUMENTS

The **reset_clock_gate_latency** command has no arguments.

DESCRIPTION

This command resets all clock-latency values previously applied to registers and clock-gating cells from all clocks that are currently defined. Only the latencies set using the **set_clock_gate_latency** command will be removed. This command will not remove latencies that were set by other sources, such as **set_clock_latency** command.

Note that this command does not remove the clock gate latency settings that were specified by the **set_clock_gate_latency** command. For this, use the **set_clock_gate_latency -reset** command.

To report the clock gate latency settings that were specified by the **set_clock_gate_latency** command, use the **report_clock_gate_latency** command.

Multicorner-Multimode Support

This command is scenario dependent and will affect the current scenario only. You must use this command on each scenario that already has a clock-gate latency setting to remove the specified configuration.

EXAMPLES

The following command removes clock-latency values from all clocks currently defined:

prompt> **reset_clock_gate_latency**

SEE ALSO

apply_clock_gate_latency(2)
compile.clockgate.physically_aware(3)
remove_clock_latency(2)
report_clock_gate_latency(2)
set_clock_gate_latency(2)
set_clock_latency(2)

reset_design

Removes all user-specified objects and attributes from the current design, except those defined by using the **set_attribute** command.

SYNTAX

status **reset_design**

ARGUMENTS

The **reset_design** command has no arguments.

DESCRIPTION

The **reset_design** command removes all user-specified modes, corner, scenarios, clocks, path groups, and constraints of the current design. Attributes defined by using the **set_attribute** command will not be removed. This command returns zero if there is no current design.

WARNING: Executing **reset_design** has wide-ranging consequences. This command removes all user-specified attributes on the design, with a result that is often equivalent to starting the design process from the beginning. If you want to reset only a few attributes, consider using the **remove_attributes** command. Alternatively, you might be able to remove selected attributes by using the commands that set them. See the appropriate man page to determine whether a specific **set_*** command has the **-default** option.

Note that when the **reset_design** command removes user-specified objects such as clocks, collections that contain those objects are implicitly deleted. For more information about collections, see the collections man page.

Multicorner-Multimode Support

This command works on all scenarios.

EXAMPLES

The following example resets the current design:

```
prompt> reset_design
```

```
1
```

The following example illustrates how the **reset_design** command can implicitly delete collections:

```
prompt> set a [get_clocks]
{clk}
prompt> query_objects $a
{clk}
prompt> reset_design
1
prompt> query_objects $a
Error: No such collection '_sel126' (SEL-001)
```

SEE ALSO

current_design(2)
remove_attributes(2)
remove_clock(2)
reset_path(2)
attributes(3)

reset_dft_configuration

Resets the DFT configuration for the current design specified by the **set_dft_configuration** command.

SYNTAX

status **reset_dft_configuration**

ARGUMENTS

The **reset_dft_configuration** command has no arguments.

DESCRIPTION

The **reset_dft_configuration** command resets the DFT configuration for the current design.

EXAMPLES

In the following example, the command resets the current design to the default configuration:

```
prompt> reset_dft_configuration
```

SEE ALSO

insert_dft(2)
report_dft_configuration(2)
set_dft_configuration(2)

reset_dft_drc_configuration

Resets the DFT DRC configuration to the default behavior.

SYNTAX

status **reset_dft_drc_configuration**

ARGUMENTS

The **reset_dft_drc_configuration** command has no arguments.

DESCRIPTION

The **reset_dft_drc_configuration** command resets any modifications to the DFT DRC configuration that were applied by the **reset_dft_drc_configuration** command.

SEE ALSO

report_dft_drc_configuration(2)
set_dft_drc_configuration(2)

reset_dft_insertion_configuration

Resets the DFT insertion configuration for the current design.

SYNTAX

status **reset_dft_insertion_configuration**

ARGUMENTS

The **reset_dft_insertion_configuration** command has no arguments.

DESCRIPTION

This command resets the existing insertion configuration to its defaults.

Use the **report_dft_insertion_configuration** command to display the current DFT insertion configuration of the design.

EXAMPLES

The following example shows how to reset the DFT insertion configuration:

```
prompt> reset_dft_insertion_configuration
```

SEE ALSO

compile(2)
insert_dft(2)
report_dft_insertion_configuration(2)
set_dft_insertion_configuration(2)

reset_disable_tie_insert

Remove the pins which are set by set_disable_tie_insert.

SYNTAX

status **reset_disable_tie_insert**
[-objects *object_name_or_collection*]
[-all]

Data Types

object_name_or_collection collection

ARGUMENTS

-objects *object_name_or_collection*

Specifies the objects for exclusion during tie cell insertion. Currently supported object types are pins and ports.If you dont specify this option then all option need to be used.

-all

Remove all the pins which are marked by set_disable_tie_insert.

DESCRIPTION

Remove the pins which are set by set_disable_tie_insert.

EXAMPLES

prompt>set_disable_tie_insert -object h1/reg_2_/SE

prompt>report_disable_tie_insert

Report:Disabled tie ports and pins

```
h1/reg_2_/SE

prompt>reset_disable_tie_insert -object h1/reg_2_/SE

prompt>report_disable_tie_insert

Report:Disabled tie ports and pins

-------------------------------------

EXAMPLE2

prompt>set_disable_tie_insert -objects [get_pins {h1/reg_2_/SE h1/reg_2_/SI}]

prompt>report_disable_tie_insert -all

Report:Disabled tie ports and pins

---------------------------------

h1/reg_2_/SE

h1/reg_2_/SI

prompt>reset_disable_tie_insert -all

prompt>report_disable_tie_insert

Report:Disabled tie ports and pins

-------------------------------------
```

SEE ALSO

add_tie_cells(2)
set_disable_tie_insert(2)
report_disable_tie_insert(2)

reset_multi_input_switching_coefficient

Resets user-specified multi-input switching (MIS) coefficients.

SYNTAX

int **reset_multi_input_switching_coefficient**
-all
object_list

Data Types

object_list list

ARGUMENTS

-all

Resets all the library cells specified for multi-input switching analysis. You cannot use this option together with the *object_list* option.

object_list

Specifies a list of library cells to be reset. You cannot use this option together with the **-all** option.

DESCRIPTION

This command resets the user-specified base coefficients for multi-input switching (MIS) analysis. If you reset the coefficient of a library cell, then the tool does not perform multi-input switching analysis on that library cell.

EXAMPLES

The following example resets the base coefficients of library cell AND2 in the library MY_LIB:

prompt> **reset_multi_input_switching_coefficient [get_lib_cells MY_LIB/AND2]**

SEE ALSO

set_multi_input_switching_coefficient(2)
report_multi_input_switching_coefficient(2)
enable_multi_input_switching_analysis(3)
enable_multi_input_switching_timing_window_filter(3)

reset_multi_vth_constraint

Disable percentage vth flow, reset low vth percentage limit and cost type to default values

SYNTAX

reset_multi_vth_constraint

DESCRIPTION

This command disables percentage vth flow and resets percentage vth constraints to default values, including low vth percentage limit and cost type. Default value of low vth percentage limit is 100.0. Default cost type is cell_count. These settings are persistent across shell sessions.

EXAMPLES

prompt> **reset_multi_vth_constraint**
Percentage vth flow : disabled
Percentage vth limit : 100.000 (%)
Percentage vth cost : cell_count

SEE ALSO

set_multi_vth_constraint(2)
report_multi_vth_constraint(2)
report_threshold_voltage_group(2)
set_vth_threshold_controls(2)

reset_parasitics_derate

Resets parasitic derating factors that influence extraction for DR/GR/CTO/RDE.

SYNTAX

string **set_parasitics_derate**
[-corners *corners*]
[-min | -max]

Data Types

corners list

ARGUMENTS

-corners *corners*

Specifies the list of corners for which to apply the extraction options. Reseting derating factors will be applied to all corners without -corners.

-min

Specifies derating factor for min analysis.

-max

Specifies derating factor for max analysis.

DESCRIPTION

This command resets derating factors to default derating factors used during RC extraction. The derating options will be associated with default constraint corner, or each list of constraint corners.

The default derating factor value is 1.0, which means no derating. This command supports 0 and positive derating factor value.

Multicorner-Multimode Support

EXAMPLES

```
prompt> report_parasitics_derate

Global factors:
---------------

Corner: default
* max
  Cap factor(Data,Clock)         : {1.1, 1.3}
  Res factor(Data,Clock)         : {1.01, 1.02}
  Coupling factor(Data,Clock)    : {1.05, 1.1}

  Layer based scale
  -----------------
    Cap scale
      M3         : 0.98
      M5         : 1.01
      M1         : 0.99
* min

  Layer based scale
  -----------------
    Cap scale
      M5         : 1.01
      M1         : 0.99
      M3         : 0.98
1
prompt> reset_parasitics_derate -max
1
prompt> report_parasitics_derate

Global factors:
---------------

Corner: default
* max
* min

  Layer based scale
  -----------------
    Cap scale
      M3         : 0.98
      M1         : 0.99
      M5         : 1.01
1
prompt> set_parasitics_derate -max -ground_capacitance_factor 1.3 -coupling_capacitance_factor 1.1 -resistance_factor 1.02 -clo
prompt> report_parasitics_derate

Global factors:
---------------
```

```
Corner: default
* max
  Cap factor(Data,Clock)         : {1, 1.3}
  Res factor(Data,Clock)         : {1, 1.02}
  Coupling factor(Data,Clock)    : {1, 1.1}
* min

  Layer based scale
  -----------------
    Cap scale
      M3         : 0.98
      M1         : 0.99
      M5         : 1.01
1
prompt> reset_parasitics_derate
1
prompt> report_parasitics_derate

Global factors:
---------------

Corner: default
* max
* min
1
```

SEE ALSO

all_corners(2)
create_corner(2)
create_mode(2)
current_corner(2)
get_corners(2)
remove_corners(2)
report_parasitics_derate(2)
set_parasitics_derate(2)
report_parasitic_parameters(2)
set_parasitic_parameters(2)
get_user_units(2)

reset_path

Resets specified paths to single-cycle behavior.

SYNTAX

```
Boolean reset_paths
  [-setup] [-hold]
  [-rise] [-fall]
  [-from from_list
    | -rise_from rise_from_list
    | -fall_from fall_from_list]
  [-through through_list]*
  [-rise_through rise_through_list]*
  [-fall_through fall_through_list]*
  [-to to_list
    | -rise_to rise_to_list
    | -fall_to fall_to_list]
  [-all]


list from_list
list rise_from_list
list fall_from_list
list through_list
list rise_through_list
list fall_through_list
list to_list
list rise_to_list
list fall_to_list
```

ARGUMENTS

-setup

Indicates that only setup (maximum delay) evaluation is to be reset to its default, single-cycle behavior. If neither **-setup** nor **-hold** is specified, both setup and hold checking are reset to single-cycle.

-hold

Indicates that only hold (minimum delay) evaluation is to be reset to its default, single-cycle behavior. If neither **-setup** nor **-hold** is specified, both setup and hold checking are reset to single-cycle.

-rise

Indicates that only rising path delays are to be reset to single-cycle behavior. If neither **-rise** nor **-fall** is specified, both rising and

falling delays are reset to single-cycle.

-fall

Indicates that only falling path delays are to be reset to single-cycle behavior. If neither **-rise** nor **-fall** is specified, both rising and falling delays are reset to single-cycle.

-from ***from_list***

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If you specify a cell, one path startpoint on that cell is affected. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-rise_from ***rise_from_list***

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from ***fall_from_list***

Same as the **-from** option, except that the path must fall from the objects specified.If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through ***through_list***

Specifies a list of pins, ports, and nets through which the paths must pass that are to be reset. Nets are interpreted to imply the leaf-level driver pins. If you omit **-through**, all timing paths specified using the **-from** and **-to** options are affected. You can specify **-through** more than once in one command invocation. For a discussion of the use of multiple **-through** options, see the DESCRIPTION section.

-rise_through ***rise_through_list***

This option is similar to the **-through** option, but applies only to paths with a rising transition at the through points. You can specify **-rise_through** more than once in one command invocation. For a discussion of the use of multiple **-through** options, see the DESCRIPTION section.

-fall_through ***fall_through_list***

This option is similar to the **-through** option, but applies only to paths with a rising transition at the through points. You can specify **-fall_through** more than once in one command invocation. For a discussion of the use of multiple **-through** options, see the DESCRIPTION section.

-to ***to_list***

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-rise_to ***rise_to_list***

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to ***fall_to_list***

Same as the **-to** option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints

clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-all

This option removes all the timing exceptions (set_false_path, set_multicycle_path, set_max_delay, set_min_delay) in this mode. This option cannot be specified with any other from/to/through or rise/fall/setup/hold option.

DESCRIPTION

The command restores designated timing paths in the current design the default single-cycle behavior. Use **reset_paths** to undo the effect of **set_multicycle_path**, **set_false_path**, **set_max_delay**, and **set_min_delay**. Attributes set by these commands are removed by **reset_paths**.

Note that a general **reset_paths** command removes the effect of matching general or specific point-to-point exception commands, as long as there is a matching object in the path specification of the original exception-setting commands. For example, the following command

reset_paths -from {CLK}

resets more specific matching exceptions for the object CLK, such as

set_false_path -from {CLK} -to {d/Z}
set_false_path -from {CLK} -to {g/Z}

If you do not specify either **-setup** or **-hold**, the default behavior is restored to both. The default is a setup relation of one cycle and a hold relation of zero cycles, so that hold is checked one active edge before the setup data at the destination register.

Setup and hold information is different for rising and falling transitions. In most cases, these are reset together. Use the **-rise** or **-fall** options to reset only one or the other. To disable setup or hold calculations for paths, use **set_false_path**.

The general and specific constraint options (for example, **-through** and **-rise_through** or **-fall_through**) are treated as different qualifiers. That is, if you issue **reset_paths -fall_through**, only those constraints are removed that were set with the **-fall_through** option; any that were set with the **-through** option are not removed. The same applies to the general and specific **-to** and **-from** options. For an example, see the EXAMPLES section.

You can use multiple **-through**, **-rise_through**, and **fall_through** options in a single command to specify paths that traverse multiple points in the design. The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

-from A1 -through B1 -through C1 -to D1

If more than one object is specified within one **-through**, **-rise_through**, or **fall_through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

-from A1 -through {B1 B2} -through {C1 C2} -to D1

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following command resets the timing relation between ff1/CP and ff2/D to the default single-cycle behavior.

prompt> **reset_paths -from { ff1/CP } -to { ff2/D }**

The following command resets only the rising delay to single cycle for all paths ending at latch2d.

prompt> **reset_paths -rise -to latch2d**

The following example first sets a specific and a general point-to-point exception, then removes the exceptions.

prompt> **set_multicycle_path 2 -from A -to Z**
prompt> **set_max_delay 15 -to Z**
prompt> **reset_paths -to Z**

In following example, the **reset_paths** command removes only the max_delay constraints, because **-to** and **-rise_to** are treated as different qualifiers and do not match.

prompt> **set_multicycle_path 2 -from A -to Z**
prompt> **set_max_delay 15 -rise_to Z**
prompt> **reset_paths -rise_to Z**

SEE ALSO

current_design(2)
reset_design(2)
set_false_path(2)
set_max_delay(2)
set_min_delay(2)
set_multicycle_path(2)

reset_paths

Resets specified paths to single-cycle behavior.

SYNTAX

Boolean **reset_paths**
[-setup] [-hold]
[-rise] [-fall]
[-from *from_list*
| -rise_from *rise_from_list*
| -fall_from *fall_from_list*]
[-through *through_list*]*
[-rise_through *rise_through_list*]*
[-fall_through *fall_through_list*]*
[-to *to_list*
| -rise_to *rise_to_list*
| -fall_to *fall_to_list*]
[-all]

list *from_list*
list *rise_from_list*
list *fall_from_list*
list *through_list*
list *rise_through_list*
list *fall_through_list*
list *to_list*
list *rise_to_list*
list *fall_to_list*

ARGUMENTS

-setup

Indicates that only setup (maximum delay) evaluation is to be reset to its default, single-cycle behavior. If neither **-setup** nor **-hold** is specified, both setup and hold checking are reset to single-cycle.

-hold

Indicates that only hold (minimum delay) evaluation is to be reset to its default, single-cycle behavior. If neither **-setup** nor **-hold** is specified, both setup and hold checking are reset to single-cycle.

-rise

Indicates that only rising path delays are to be reset to single-cycle behavior. If neither **-rise** nor **-fall** is specified, both rising and

falling delays are reset to single-cycle.

-fall

Indicates that only falling path delays are to be reset to single-cycle behavior. If neither **-rise** nor **-fall** is specified, both rising and falling delays are reset to single-cycle.

-from ***from_list***

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If you specify a cell, one path startpoint on that cell is affected. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-rise_from ***rise_from_list***

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from ***fall_from_list***

Same as the **-from** option, except that the path must fall from the objects specified.If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through ***through_list***

Specifies a list of pins, ports, and nets through which the paths must pass that are to be reset. Nets are interpreted to imply the leaf-level driver pins. If you omit **-through**, all timing paths specified using the **-from** and **-to** options are affected. You can specify **-through** more than once in one command invocation. For a discussion of the use of multiple **-through** options, see the DESCRIPTION section.

-rise_through ***rise_through_list***

This option is similar to the **-through** option, but applies only to paths with a rising transition at the through points. You can specify **-rise_through** more than once in one command invocation. For a discussion of the use of multiple **-through** options, see the DESCRIPTION section.

-fall_through ***fall_through_list***

This option is similar to the **-through** option, but applies only to paths with a rising transition at the through points. You can specify **-fall_through** more than once in one command invocation. For a discussion of the use of multiple **-through** options, see the DESCRIPTION section.

-to ***to_list***

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-rise_to ***rise_to_list***

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to ***fall_to_list***

Same as the **-to** option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints

clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-all

This option removes all the timing exceptions (set_false_path, set_multicycle_path, set_max_delay, set_min_delay) in this mode. This option cannot be specified with any other from/to/through or rise/fall/setup/hold option.

DESCRIPTION

The command restores designated timing paths in the current design the default single-cycle behavior. Use **reset_paths** to undo the effect of **set_multicycle_path**, **set_false_path**, **set_max_delay**, and **set_min_delay**. Attributes set by these commands are removed by **reset_paths**.

Note that a general **reset_paths** command removes the effect of matching general or specific point-to-point exception commands, as long as there is a matching object in the path specification of the original exception-setting commands. For example, the following command

reset_paths -from {CLK}

resets more specific matching exceptions for the object CLK, such as

set_false_path -from {CLK} -to {d/Z}
set_false_path -from {CLK} -to {g/Z}

If you do not specify either **-setup** or **-hold**, the default behavior is restored to both. The default is a setup relation of one cycle and a hold relation of zero cycles, so that hold is checked one active edge before the setup data at the destination register.

Setup and hold information is different for rising and falling transitions. In most cases, these are reset together. Use the **-rise** or **-fall** options to reset only one or the other. To disable setup or hold calculations for paths, use **set_false_path**.

The general and specific constraint options (for example, **-through** and **-rise_through** or **-fall_through**) are treated as different qualifiers. That is, if you issue **reset_paths -fall_through**, only those constraints are removed that were set with the **-fall_through** option; any that were set with the **-through** option are not removed. The same applies to the general and specific **-to** and **-from** options. For an example, see the EXAMPLES section.

You can use multiple **-through**, **-rise_through**, and **fall_through** options in a single command to specify paths that traverse multiple points in the design. The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

-from A1 -through B1 -through C1 -to D1

If more than one object is specified within one **-through**, **-rise_through**, or **fall_through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

-from A1 -through {B1 B2} -through {C1 C2} -to D1

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following command resets the timing relation between ff1/CP and ff2/D to the default single-cycle behavior.

prompt> **reset_paths -from { ff1/CP } -to { ff2/D }**

The following command resets only the rising delay to single cycle for all paths ending at latch2d.

prompt> **reset_paths -rise -to latch2d**

The following example first sets a specific and a general point-to-point exception, then removes the exceptions.

prompt> **set_multicycle_path 2 -from A -to Z**
prompt> **set_max_delay 15 -to Z**
prompt> **reset_paths -to Z**

In following example, the **reset_paths** command removes only the max_delay constraints, because **-to** and **-rise_to** are treated as different qualifiers and do not match.

prompt> **set_multicycle_path 2 -from A -to Z**
prompt> **set_max_delay 15 -rise_to Z**
prompt> **reset_paths -rise_to Z**

SEE ALSO

current_design(2)
reset_design(2)
set_false_path(2)
set_max_delay(2)
set_min_delay(2)
set_multicycle_path(2)

reset_pipeline_scan_data_configuration

Resets the pipeline scan data configuration specified by the **set_pipeline_scan_data_configuration** command.

SYNTAX

status **reset_pipeline_scan_data_configuration**

ARGUMENTS

The **reset_pipeline_scan_data_configuration** command has no arguments.

DESCRIPTION

This command resets the pipeline scan data configuration for the current design.

EXAMPLES

The following example resets the pipeline scan data configuration of the current design to the default:

prompt> **reset_pipeline_scan_data_configuration**

SEE ALSO

set_pipeline_scan_data_configuration(2)

reset_placement

Unplace all cells in the core area

SYNTAX

collection **reset_placement**
[-spread_cells]

-spread_cells

Specifies that the cells should be spread out to the right of the block.

DESCRIPTION

Unplace cells in the core area. The command does not unplace fixed & dont_touch cells. I

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example unplaces all cells

prompt> **reset_placement**

SEE ALSO

create_placement(2)

reset_placement_ir_drop_target

Reset placer IR drop targets

SYNTAX

string **reset_placement_ir_drop_target** *voltage_area*
type

list *voltage_area*
string *type*

ARGUMENTS

voltage_area

Specifies the voltage area for settings the IR drop targets

type

Specifies the type of IR drop target setting, either low or high

DESCRIPTION

This command resets the IR aware placement settings.

EXAMPLES

The followin example sets for the voltage area DEFAULT_VA the low voltage drop target to 3.0% of the effective supply voltage and the high voltage drop target to 5.0% of the effective supply voltage. Then the high target is reset.

```
prompt> set_placement_ir_drop_target DEFAULT_VA low 3.0 -irdrop
prompt> set_placement_ir_drop_target DEFAULT_VA high 5.0 -irdrop
prompt> get_placement_ir_drop_target DEFAULT_VA
{ { DEFAULT_VA low 3.000000 -irdrop } { DEFAULT_VA high 5.000000 -irdrop } }
prompt> reset_placement_ir_drop_target DEFAULT_VA high
prompt> get_placement_ir_drop_target DEFAULT_VA
{ { DEFAULT_VA low 3.000000 -irdrop } }
```

SEE ALSO

set_placement_ir_drop_target(2)
get_placement_ir_drop_target(2)
report_placement_ir_drop_target(2)
place.coarse.ir_drop_default_target_low_population(3)
place.coarse.ir_drop_default_target_low_percentage(3)
place.coarse.ir_drop_default_target_high_population(3)
place.coarse.ir_drop_default_target_high_percentage(3)

reset_power_budget

Resets power budget set on the design, or on the hierarchical instances in the design.

SYNTAX

status **reset_power_budget**
[-scenarios *scenario_list*]
[-cell*cell_list*]

Data Types

scenario_list list
cell_list list

ARGUMENTS

-scenarios *scenario_list*

Indicates the scenario for which the budget values are to be reset. If no scenario is specified, the current_scenario is used.

-cell*cell_list*

Specifies the list of cells on which the power budget values are reset.

DESCRIPTION

Invoke this command with no arguments to reset all budget. This includes both budget values set globally on the design and those set on specific instances in the current design. If the command is called with a list of cells, only the power budget values on the specified cells are reset. Note that power budget value set specifically on a cell is not overwritten by a set or reset command on its parent hierarchical cell. To reset the power budget values set specifically on the child cells, include them in the cell list with the -cell option.

Multicorner-Multimode Support

EXAMPLES

The following command resets both power budget values set globally and those set on specific instances in the design for the

scenario SC1.

```
prompt> reset_power_budget -scenario SC1
```

The following example resets the power budget set on cell H1 for the current scenario.

```
prompt> reset_power_budget -cell [get_cell H1]
```

The following command resets the power budget values set on all cells matching "*"

```
prompt> reset_power_budget -cell [get_cells *]
```

Assuming that the hierarchical cell H1 contains a cell H2, then the following command resets the budget set on H2.

```
prompt> reset_power_budget -cell [get_cells H1/H2]
```

SEE ALSO

set_power_budget(2)
get_power_budget(2)
report_power_budget(2)
report_power(2)

reset_power_clock_scaling

Resets the clock frequency scaling data for power analysis.

SYNTAX

status **reset_power_clock_scaling**
[-scenarios *scenario_list*]
[-all]
[*clock_list*]

Data Types

scenario_list list
clock_list list

ARGUMENTS

-scenarios *scenario_list*

Specify the list of scenarios for which the scaling data will be reset. If not specified, the current scenario is used.

clock_list

Specify a list of clocks in the design for which the scaling data is to be reset. If this option is not used then the global ratio value for the scenario is reset. If this option is used then the clock specific value for the scenario will be reset.

-all

A boolean option to clear the entire scaling data present for all the scenarios.

DESCRIPTION

The command resets the scaling data already present for the design. If '-all' option is used, then the entire scaling data is removed. The '-scenarios' option determines which scenario specifc data is to be reset. If the option is not used, the current scenario is the scenario taken for reset operation. The 'clock_list' option will reset the clock specific data for the scenario(s) specified or the current scenario if '-scenarios' option is not used. If the option is not used, then the global ratio value will be reset for the applicable scenario(s). The command returns 1 on success and 0 on failure.

EXAMPLES

The following examples describe the behavior of the cmd.

The following example resets the clk1 data present for the current scenario.

```
prompt> reset_power_clock_scaling clk1
```

The following example resets the clk1 data for the scenario scn1 having mode scn1.mode.

```
prompt> reset_power_clock_scaling [get_clocks -mode scn1.mode clk1]
```

The following example resets the global ratio values for the scenarios 'default' and 'scn1'

```
prompt> reset_power_clock_scaling -scenarios {default scn1}
```

The following example resets the clk1 and clk2 data for the scenario scn1 having mode scn1.mode, and global ratio value for the scenario 'default'.Here the current scenario is scn1.

```
prompt> reset_power_clock_scaling -scenarios {scn1 default} {clk1 clk2}
```

SEE ALSO

get_power_clock_scaling(2)
set_power_clock_scaling(2)

reset_power_derate

Resets the power derating factors for either the current design, a list of cells, library cells or all cells in a power group in the current design.

SYNTAX

status **reset_power_derate**
[-scenarios *scenario_list*]
[-groups *group_names*]
[*object_list*]

Data Types

scenario_list list
group_names list
object_list list

ARGUMENTS

-scenarios *scenario_list*

Indicates the scenario for which the derate_value is to be reset. If no scenario is specified the current_scenario is used.

-group *group_names*

Specifies the power group for which the derate factor is to be reset. The derate factors are reset for each cell object in the power group. Note groups and object_list are mutually exclusive option.

object_list

Specifies the list of cells or library cells on which the power derating factors are reset. Note groups and object_list are mutually exclusive option.

DESCRIPTION

Invoke this command with no arguments to reset all derate (to the default of 1.0). This includes both derating factors set globally on the design and those set on specific instances in the current design. If the command is called with a list of objects, only the power derating factors on the specified objects are reset. After a reset_power_derate command is applied to an object, the power derating factors of all power components, including leakage, internal and switching power are reset.

Note that power derating factors set specifically on an object are not overwritten by a set or reset command on its parent hierarchical cell. To reset the power derating factors set specifically on the child cells, include them in the object list of the command.

The -group options allows to reset the derate factors of cells only in the specified group.

Heterogeneous collections of objects can be combined and passed to this command.

This command does not impact the power budget applied using set_power_budget.

Multicorner-Multimode Support

EXAMPLES

The following command resets both power derating factors set globally and those set on specific instances in the design for the scenario SCN1.

```
prompt> reset_power_derate -scenario SCN1
```

The following example resets the power derating factors set on cell U1 for the current scenario.

```
prompt> reset_power_derate [get_cell U1]
```

The following command resets the power derating factors set on all cells matching "*"

```
prompt> reset_power_derate [get_cells *]
```

The following example resets the power derating factors set on all instances of library cell IV in the library MY_LIB.

```
prompt> reset_power_derate [get_lib_cells MY_LIB/IV]
```

Assuming that the hierarchical cell H1 contains a cell U1, then the following command resets the derate factors set on U1. As a result, U1 inherits the derate factors that are set on H1.

```
prompt> reset_power_derate [get_cells H1/U1]
```

SEE ALSO

set_power_derate(2)
get_power_derate(2)
report_power_derate(2)

reset_power_group

Resets user specified power group on cells and libcells.

SYNTAX

status **reset_power_group**
object_list
-name *group_name*
-all

Data Types

object_list list
group_name string

ARGUMENTS

-name *group_name*

Specifies the user power group name that needs to be removed on all the cells and libcells present in the design having that power group name.

object_list

Specifies list of cells and/or libcells whose user specified power groups needs to be removed.

-all

Resets user specified power group of all the cells and libcells in the design.

DESCRIPTION

The command resets user specified power group(s). One and only one option among '-name', 'object_list' or '-all' can be specified.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example resets power group 'macro1' from all libcells and cells having that power group name.

```
prompt> reset_power_group -name macro1
```

SEE ALSO

get_power_group(2)
set_power_group(2)
report_power_groups(2)
get_power_group_objects(2)
report_power(2)
power.report_user_power_groups(3)

reset_pvt

Resets all user specified process, voltage, temperature, and derate settings set on one or more corners of the design.

SYNTAX

int **reset_pvt**
[-corners *corner_list*]

list *corner_list*

ARGUMENTS

-corners *corner_list*

Specifies a list of corners to be processed. If this option is not given, the current corner will be processed.

DESCRIPTION

This command will reset all process label, process number, voltage, temperature, and timing derate settings for the given corner(s). Voltages set on UPF supply nets and supply ports will also be reset. All timing derates will be reset to 1.0. After this command is executed, the corner(s) will have the same settings that a newly-created corner would have, based on the main library default PVT values.

Multicorner-Multimode Support

By default, this command works on the current corner. To specify a different corner, use the **-corners** option.

SEE ALSO

set_process_label(2)
set_process_number(2)
set_process_voltage(2)
set_process_temperature(2)
set_timing_derate(2)

reset_scan_configuration

Resets the scan configuration for the current design.

SYNTAX

status **reset_scan_configuration**

ARGUMENTS

The **reset_scan_configuration** command has no arguments.

DESCRIPTION

The **reset_scan_configuration** command resets the current scan configuration from the design.

EXAMPLES

The following example shows how to reset the scan configuration:

```
prompt> reset_scan_configuration
```

SEE ALSO

report_scan_configuration(2)
set_scan_configuration(2)

reset_supply_net_probability

Resets switching probability on selected supply nets in the current design.

SYNTAX

status **reset_supply_net_probability**
[-scenarios *scenario_list*]
[-modes *mode_list*]
[-corners *corner_list*]
object_list

Data Types

scenario_list	list
mode_list	list
corner_list	list
object_list	list

ARGUMENTS

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Probability is reset separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, probability is reset for the current scenario in the design.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering on modes will occur.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If this option is not specified, no filtering on corners will occur.

DESCRIPTION

Use this command to reset the switching activity of design supply nets. The activity of a supply net is used to scale leakage power and dynamic power of the cells connected to the supply net. The scaling of leakage power is governed by app option *power.scale_leakage_power_at_power_off* (by default true), the scaling of dynamic power by app option

power.scale_dynamic_power_at_power_off (by default false). The scaling factor for a cells is the lowest static_probability of all its supply nets.

The supply nets that are being reset can be specified explicitly as a list of objects.

If a combination of a mode and corner is specified that does not represent a valid scenario, those are ignored.

Multicorner-Multimode Support

EXAMPLES

The following **reset_supply_net_probablity** command reset the static_probablity on a supply net

```
prompt> set_supply_net_probability -static_probability 0.9 [get_supply_net VDD]
```

SEE ALSO

set_supply_net_probability(2)
get_supply_net_probiblity(2)
power.scale_leakage_power_at_power_off(3)
power.scale_dynamic_power_at_power_off(3)

reset_switching_activity

Resets switching activity information on nets, pins, ports and cells in the current design.

SYNTAX

status **reset_switching_activity**
[-state_condition *state_condition*]
[-path_sources *path_sources*]
[-scenarios *scenario_list*]
[-modes *mode_list*]
[-corners *corner_list*]
[*object_list*]

Data Types

state_condition	string
path_sources	list
scenario_list	list
mode_list	list
corner_list	list
object_list	list

ARGUMENTS

-state_condition *state_condition*

Specifies the state condition to reset state-dependent activities. State dependent activities can be reset when the object is already annotated with state-dependent activities. The state condition specified with this argument must match to a state condition already annotated for this object. Use -state_condition default to specify the default state condition.

-path_sources *path_sources*

Specifies the path sources to reset path-dependent activities. This can be used when the object is already annotated with path-dependent activities. The path sources specified with this argument must be the same as those annotated for this object.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Activity is reset separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, activity is reset for the current scenario in the design.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering on modes will occur.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If this option is not specified, no filtering on corners will occur.

object_list

Specifies a list of nets, pins or ports in the current design for which the values of static probability, rise toggle rate, fall toggle rate and type of activity are reset.

DESCRIPTION

Use this command to reset switching activity of nets, ports, pins and cells in the design. The values reset include rise toggle rate, fall toggle rate, static probability and the type of activity.

The design objects whose switching activities are reset can be specified as a list of objects. If no such list is provided, activities of all objects in the design which had been annotated with the **set_switching_activity** command will be reset.

If an instance is specified in the object list, then the state-dependent path_dependent activity information will be removed from the instance.

Multicorner-Multimode Support

EXAMPLES

The following example depicts the command resetting the values of rise toggle rate, fall toggle rate, static probability and type of activity on the pin 'InstDecode/InstReg_reg_23_/SI'. Initially the pin had the following activity:

(mode = default, corner = default, probability = 0.9, rise_toggle_rate = 0.4, fall_toggle_rate = 0.6, type = annotated)

After reset, it will have the following activity.

(mode = default, corner = default, probability = 0.8, rise_toggle_rate = 0.116667, fall_toggle_rate = 0.116667, type = default)

Such default activity would be derived from the values as specified by the user in the app options 'power_default_static_probability', 'power_default_toggle_rate' and 'power_default_toggle_rate_reference_clock'.

```
prompt> reset_switching_activity {InstDecode/InstReg_reg_23_/SI}
```

SEE ALSO

get_switching_activity(2)
read_saif(2)
report_power(2)
report_switching_activity(2)
set_switching_activity(2)

reset_test_mode

Resets the test mode for the current design.

SYNTAX

status **reset_test_mode**

ARGUMENTS

The **reset_test_mode** command has no arguments.

DESCRIPTION

The **reset_test_mode** command resets the current test mode of the current design to the default test mode.

EXAMPLES

The following example resets the current test mode:

```
prompt> reset_test_mode
Current test mode is reset to default
1
```

SEE ALSO

current_test_mode(2)
define_test_mode(2)

reset_testability_configuration

Resets the testability configuration for the current design.

SYNTAX

status **reset_testability_configuration**

ARGUMENTS

The **reset_testability_configuration** command has no arguments.

DESCRIPTION

The **reset_testability_configuration** command resets the testability configuration for the current design.

SEE ALSO

report_testability_configuration(2)
set_testability_configuration(2)

reset_timing_derate

Resets user specified derate factors set either on a design or on a specified list of objects (cells, library cells, or nets).

SYNTAX

int **reset_timing_derate**
[-hierarchical_net_delay]
[-scalar] [-variation]
[-aocvm_guardband]
[-pocvm_guardband]
[-pocvm_coefficient_scale_factor]
[-pocvm_subtract_sigma_factor_from_nominal]
[-increment]
[-corners *corner_list*]
object_list

list *object_list*
list *corner_list*

ARGUMENTS

-hierarchical_net_delay

Resets net delay derating (as well as cell delay derating) previously set by the **set_timing_derate** command for the hierarchical cells specified in the *object_list*. Without the **-hierarchical_net_delay** option, the **reset_timing_derate ... [get_cells ...]** command resets only the cell delay (not net delay) derating in the hierarchical cells.

-scalar

Indicates that only derate factors for deterministic delays should be reset. This option is currently ignored.

-variation

Indicates that only derate factors for statistical delays should be reset. This option is currently ignored.

-aocvm_guardband

Indicates that only AOCVM guardband derate factors should be reset.

-pocvm_guardband

Indicates that only POCVM guardband derate factors should be reset.

-pocvm_coefficient_scale_factor

Indicates that only POCVM coefficient scale factors should be reset.

-pocvm_subtract_sigma_factor_from_nominal

Indicates that only POCVM subtract sigma factors should be reset.

-increment

Indicates that only incremental derate factors for the specifed delays should be reset.

-corners *corner_list*

Specifies the corner(s) where the derate values will be reset. If this option is not given, the current corner will be used.

object_list

Specifies a list of designs, cells, library cells or nets which will be reset.

DESCRIPTION

Call this command with no arguments to reset all derate factors set on the design (to the default value, 1.0). This will include both derate factors set globally on the design and those set on specific instances in the design. Call this command with a list of objects to reset a specific set of objects.

The *object_list* option may be used to reset derate factors on specific objects (cells or library cells) in the design. All derate factors are reset on each instance. If *object_list* contains a design, then only global derate factors are reset and instance specific derate factors are not reset.

Heterogeneous collections of objects may be combined and passed to this command.

If neither *-scalar* nor *-variation* options are specified, then both deterministic and variation derates are reset.

The incremental derate factors are reset when no options are specified or when -increment option is specified.

If *object_list* contains any hierarchical cells then all cells within that hierarchical cell and also all cells of all lower hierarchies will have their inherited derate factors updated accordingly. Note that derate factors set specifically on an object will not be overwritten by a set/reset command on its parent hierarchical cell.

Multicorner-Multimode Support

By default, this command works on the current corner. To specify a different corner, use the **-corners** option.

EXAMPLES

The following command resets both incremental and non-incremental derate factors set globally on the design and those set on specific instances in the design.

prompt> **reset_timing_derate**

The following command resets the derate factors set globally on the design MY_DESIGN only.

prompt> **reset_timing_derate [get_designs MY_DESIGN]**

The following example resets the derate factors set on cell U1.

prompt> **reset_timing_derate [get_cells U1]**

The following command resets the derate factors set on all cells matching "*".

prompt> **reset_timing_derate [add_to_collection [get_cells *]]**

The following example resets the derate factors set on all instances of library cell IV in the library MY_LIB.

prompt> **reset_timing_derate [get_lib_cells MY_LIB/IV]**

Assuming that the hierarchical cell H1 contains a cell U1, then the following command resets the derate factors set on U1 and as a result U1 inherits the derate factors that are set on H1.

prompt> **reset_timing_derate [get_cells H1/U1]**

Assuming that the hierarchical cell H1 also contains a hierarchical cell H2, then the following command resets the derate factors set on H2 and as a result H2 and all of its cells inherit the derate factors that are set on H1.

prompt> **reset_timing_derate [get_cells H1/H2]**

Following example reset only incremental derates on cell top/H1/u12.

prompt> **reset_timing_derate -increment [get_cells top/H1/u12]**

The following command resets the cell **and net** delay derating previously set on hierarchical cell H1/H2. As a result, cell H2 and all of its lower-level cells inherit the cell **and net** delay derating set on their parent (or grandparent) cell, H1.

prompt> **reset_timing_derate -hierarchical_net_delay [get_cells H1/H2]**

SEE ALSO

set_timing_derate(2)
report_timing_derate(2)

reset_upf

Removes all UPF constraints and UPF data-dependent constraints from the current design. The entire design is restored back to default single voltage UPF condition.

SYNTAX

status **reset_upf**

ARGUMENTS

The **reset_upf** command has no arguments.

DESCRIPTION

The **reset_upf** command is used to clean up all UPF constraints in the current design. This command returns a 1 if successful and returns a 0 otherwise.

The **reset_upf** command also removes UPF data-dependent constraints, which include the following commands:

set_voltage
set_related_supply_net

After **reset_upf** is executed and new UPF constraints are reloaded using the **load_upf** command, run the above commands again with respect to the new UPF constraints.

Warning: Executing **reset_upf** has extremely wide-ranging consequences. This command removes all UPF constraints in the current design, with a result that is often equivalent to starting the UPF flow. This command removes UPF constraints without touching the netlist. The command does not check if the netlist has been processed by prior UPF constraints. All commands that might change the netlist after the **load_upf** command will not be backed out by the **reset_upf** command. Remember to verify that the UPF constraints are consistent.

EXAMPLES

The following example shows a typical use of the *reset_upf* command:

```
prompt> load_upf upf_file1
prompt> reset_upf
```

```
prompt> load_upf upf_file2
1
```

The following example shows one of the risks of using the **reset_upf** command. After running the **reset_upf** command, the power domain is removed along with the relationship of the power domain with its related voltage area. So even after reloading the UPF file, you cannot get related power domain information on those voltage areas. In this case, remove the related voltage areas and recreate them with the power domain.

```
prompt> get_voltage_area VA1
VA1
prompt> get_power_domain VA1
VA1
prompt> reset_upf
prompt> load_upf upf_file
prompt> report_voltage_areas VA1
```

SEE ALSO

load_upf(2)
save_upf(2)

reset_via_ladder_candidates

Resets all via ladder candidates for the specified library-pin.

SYNTAX

status **reset_via_ladder_candidates**
lib_pin

Data Types

lib_pin pin name or collection

ARGUMENTS

lib_pin

Specifies the library-pin for which via ladder candidate will be reset.

DESCRIPTION

This command clears all via ladder candidates for the specified library pin. This command returns 1 if succeeded, 0 otherwise.

EXAMPLES

The following example resets via ladder candidates for specified pins.

```
prompt> reset_via_ladder_candidates [get_lib_pin tcbn90ghvt/AN2HVTD0/A1]
1
```

SEE ALSO

set_via_ladder_candidate(2)
report_via_ladder_candidates(2)
set_via_ladder_constraints(2)

remove_via_ladder_constraints(2)
report_via_ladder_constraints(2)

reset_wrapper_configuration

Resets the wrapper configuration for the current design.

SYNTAX

status **reset_wrapper_configuration**
[-test_mode *mode_name*]

Data Types

mode_name string

ARGUMENTS

-test_mode *mode_name*

Specifies the test mode to which the **reset_wrapper_configuration** command applies. The default is **all**, which resets the global wrapper configuration, but keeps any mode-specific configurations.

DESCRIPTION

This command resets the wrapper configuration to the following values:

```
Wrapper Configuration
====================
Reuse Threshold:                  0
Hier Wrapping:                    Disable
Add Wrapper Cells To Power Domain:    Disable
Mix Cells:                        False
Mix With Scan Cells:              False
Use System Clock For Dedicated Wrapper Cells: Enable
Depth Threshold:                  Unspecified
Chain count:                      Unspecified
Gate Cells:                       none
```

SEE ALSO

insert_dft(2)
preview_dft(2)
report_wrapper_configuration(2)
set_wrapper_configuration(2)

reshape_objects

Cuts from the boundary or adds to the boundary of one or more geometric objects.

SYNTAX

string **reshape_objects**
-add *rectangle*
-cut *rectangle*
-cut_by_locked
[-ignore_end_cap]
[-gap *distance*]
[-gap_min_spacing]
[-keep_inside]
[-via_cut_pattern]
[-force]
[-simple]
[*object_list*]

Data Types

rectangle	rectangle string
object_list	collection or selection
distance	float

ARGUMENTS

object_list

Collection or selection set containing objects to reshape. If this option is not specified, global selection set is used.

-add *rectangle*

Specifies the rectangle to append to the object boundaries.

-cut *rectangle*

Specifies the rectangle to cut from the object boundaries.

-cut_by_locked

Cuts object boundary by another locked object. It implicitly works on the specified objects. The objects must either be resizable or support a rectilinear boundary depending on the final result of the cut.

-ignore_end_cap

Treats the wire as if it has no cap extension.

-gap *distance*

Specifies an extra gap spacing by which to cut. This option is mutually exclusive with the **-gap_min_spacing**. By default, the gap spacing is 0 if neither **-gap_min_spacing** nor **-gap** distance are specified.

-gap_min_spacing

Specifies a gap spacing as the minimum spacing rule. This option is mutually exclusive with the **-gap distance** option. By default, the gap spacing is 0 if neither **-gap_min_spacing** nor **-gap** distance are specified.

-keep_inside

Controls whether the tool removes the parts of the object outside the rectangle and keeps the inside parts. This option works only with the **-cut** option.

-via_cut_pattern

Updates via cut pattern of multi cut via. This option works with both the **-add** and the **-cut** options. By default, multi cut vias can only be trimmed. Note that the term "cut" refers to via metal pieces on the middle layer.

-force

Resizes locked or fixed objects. By default, such objects are not resized according to global edit settings. Edit settings can be accessed using **get_edit_setting**.

-simple

Disable snapping and editing constraints. By default, objects are snapped according to global snap settings. Snap settings can be accessed using **get_snap_setting**.

DESCRIPTION

This command cuts from or adds to the boundary of a set of objects by using an explicit rectangle or the boundary of fixed objects.

When multiple specified objects are to be cut, each object is treated independently and the result is the cumulative result of all cuts.

Note that cutting takes place only if the rectangle, or fixed object boundary by which to cut, overlaps with the boundary of the object being cut.

Snapping is done automatically by using global snap settings.

The command returns a collection of the reshaped objects if the input *object_list* is a collection of objects. Otherwise, this command reshapes objects in the specified selection set and returns status.

EXAMPLES

The following example cuts the boundary of the selected objects by a bounding box defined by the points {0 0} {100 100}.

```
prompt> reshape_objects -cut {{0 0} {100 100}}
```

The following example cuts the boundary of three the non-fixed selected objects by one fixed selected objects:

```
prompt> sizeof_collection [get_selection] "is_fixed==false"]
3
```

```
prompt> sizeof_collection [get_selection] "is_fixed==true"]
1
prompt> reshape_objects -cut_by_locked
```

The alternative syntax uses collection to specify objects.

```
prompt> reshape_objects [get_selection] -cut_by_locked
```

SEE ALSO

change_selection(2)
get_selection(2)
get_snap_setting(2)
set_object_shape(2)
set_snap_setting(2)
snap_objects(2)

resize_objects

Resizes one or more objects. Resizes the width and height of move bounds, keepouts, and obstruction objects.

SYNTAX

status **resize_objects**
{ -bbox *bounding_box* |
-delta *offset* |
-scale *scale_factor* |
-width *object_width* |
-height *object_height* |
-utilization *util_factor* |
-aspect *aspect* |
-area *area* }
[-force]
[-simple]
[-quiet]
[*object_list*]

Data Types

bounding_box rectangle
offset rectangle
scale_factor point
object_width real
object_height real
util_factor real
aspect real
area real
object_list collection

ARGUMENTS

object_list

Collection or selection set containing objects to reshape. If this option is not specified, global selection set is used.

-bbox *bounding_box*

Specifies the new bounding box of the object.

The **-bbox** option cannot be used with any of the following options: **-delta**, **-scale**, **-width**, **-height**, **-utilization**, **-aspect** and **-area**.

-delta *offset*

Specifies the offset of the current bounding box of the object.

The *offset* argument should be of the following formats: {{x1 y1} {x2 y2}}.

The **-delta** option cannot be used with any of the following options: **-bbox**, **-scale**, **-width**, **-height**, **-utilization**, **-aspect** and **-area**.

-scale ***scale_factor***

Specifies an x-axis and y-axis scale factor. All objects will be resized by this factor.

The **-scale** option cannot be used with any of the following options: **-bbox**, **-delta**, **-width**, **-height**, **-utilization**, **-aspect** and **-area**.

The scale factor must be positive and non-zero.

-width ***object_width***

Specifies the width for a given object (or collection of objects).

All objects are resized so that their bounding box has this width and lower-left corner or center does not change. For pins located on the edge of a block, the width is a dimension parallel to the edge.

The **-width** option cannot be used with any of the following options: **-bbox**, **-delta**, **-scale**, **-utilization**, **-aspect** and **-area**.

The **-width** and **-height** options can be used together.

-height ***object_height***

Specifies the height for the given object (or collection of objects).

All objects are resized so that their bounding box has this width and lower-left corner or center does not change. For pins located on the edge of a block, the height is a dimension orthogonal to the edge.

The **-height** option cannot be used with any of the following options: **-bbox**, **-delta**, **-scale**, **-utilization**, **-aspect** and **-area**.

The **-width** and **-height** options can be used together.

-utilization ***util_factor***

Specifies the required utilization for the object. The object changes size based on *util_factor* value, but keeps the lower-left corner (center for pins and vias) fixed and retains the original aspect ratio.

The **-utilization** option cannot be used with any of the following options: **-bbox**, **-delta**, **-scale**, **-width**, **-height**, **-aspect** and **-area**.

The utilization factor must be positive and non-zero.

Note: this option is usable only for soft macros and movebounds.

-aspect ***aspect***

Specifies the required aspect for the object. The object changes size based on the *aspect* value, but keeps the lower-left corner (center for pins and vias) fixed and retains the original area.

The **-aspect** option cannot be used with any of the following options: **-bbox**, **-delta**, **-scale**, **-width**, **-height**, **-utilization** and **-area**.

-area ***area***

Specifies the required area for the object.

The **-area** option cannot be used with any of the following options: **-bbox**, **-delta**, **-scale**, **-width**, **-height**, **-utilization** and **-aspect**.

-force

Resizes locked or fixed objects. By default, such objects are not resized according to global edit settings. Edit settings can be

accessed using **get_edit_setting**.

-simple

Disable snapping and editing constraints. By default, objects are snapped according to global snap settings. Snap settings can be accessed using **get_snap_setting**.

-quiet

Suppress all warning messages.

DESCRIPTION

This command resizes a list of objects to either an explicit bounding box, a delta from the current bounding box, an x and y scale, an explicit width and height, a size that gives a required utilization given a constant aspect ratio, or a size that gives a specified aspect ratio given a constant area.

Only resizable objects can be used with this command.

Notes

Locked objects are not moved unless you use the **-ignore_locked** option.

Snapping is done automatically using global snap settings. This command can be used to adjust the width and height of objects specified in the object list. You can use it to adjust the width and height of a single object or apply it to multiple objects that are individually resized and respaced in a linearly.

EXAMPLES

The following example resizes all selected objects so that they have an aspect ratio of 2.5.

prompt> **resize_objects [get_selection] -aspect 2.5**

The following example resizes all specified objects so that their lower-left corner is at the point {0 0} and they are 100 units wide and 100 units high.

prompt> **resize_objects [get_selection] -bbox {{0 0} {100 100}}**

SEE ALSO

change_selection(2)
get_selection(2)
get_snap_setting(2)
set_snap_setting(2)
set_object_shape(2)
snap_objects(2)

resize_polygons

Adjusts the edges of a geometric region and returns the result as a geo_mask.

SYNTAX

collection **resize_polygons**
[-objects *object_list*]
[*pos_object_list*]
-size { *d* | *lr tb* | *l b r t* }

Data Types

object_list collection
pos_object_list collection
d float
lr float
tb float
l float
b float
r float
t float

ARGUMENTS

-objects *object_list*

Specifies the objects to be used to define the geometric region to be resized. Objects may be a heterogenous collection of poly_rects, geo_masks, shapes, layers, and other physical objects.

In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

The specified region must not be empty, i.e. must have positive area. It is an error to attempt to resize an empty region.

pos_object_list

Specifies the objects to be used to define the geometric region to be resized. This positional option is provided for compatibility with other tools.

-size { *d* | *lr tb* | *l b r t* }

Specifies the distance to expand the edges. If one value is provided, the left, bottom, right, and top edges are each expanded by that distance. If two values are provided, the left and right edges are each expanded by the first distance, and the top and bottom

edges are each expanded by the second distance. If four values are provided, the left, bottom, right, and top edges are expanded by the respective distances.

You may specify negative distance values, in which case the respective edges will be shrunk by the absolute value. Zero value distances will result in no change.

This is a required option.

DESCRIPTION

This command expands or shrinks the boundaries of the region specified by *objects*, and returns a collection that contains the result as geo_mask object.

EXAMPLES

The following example returns a geo_mask with a region that is a shape padded by 5 units of distance.

prompt> **resize_polygons -objects [get_shapes RECT_0] -size {5}**

The following example returns a geo_mask with a region composed of the shapes on a layer, with each contiguous polygon expanded by a margin of 15 units of distance on its right-hand side.

prompt> **resize_polygons -objects [get_layers M1] -size {0 0 15 0}**

SEE ALSO

copy_to_layer(2)
create_poly_rect(2)
create_geo_mask(2)
compute_polygons(2)
split_polygons(2)
compute_area(2)
transform_polygons(2)

resolve_pg_nets

Resolves power and ground connections based on power domains.

SYNTAX

status **resolve_pg_nets**
[-check_only]
[-design *design*]
[-verbose]

ARGUMENTS

-check_only

Specifies the check_only mode. Under this mode, the **resolve_pg_nets** command prints out intended PG connection changes without applying the changes.

-design *design*

Specifies the design in which to resolve the power and ground nets. If no design is specified, the power and ground nets are resolved in the current design.

-verbose

Prints the details of the changes to the power and ground connections.

DESCRIPTION

This command resolves the conflicts between power and ground network with power domains. In MV mode, UPF must be committed before running the **resolve_pg_nets** command. In non MV mode, the default power domain is used to resolve PG connections.

EXAMPLES

The following example uses the **resolve_pg_nets** command to resolve PG connection based on power domains.

prompt> **resolve_pg_nets -verbose**

The following example uses the **resolve_pg_nets** command to check PG connection against power domains without making any changes.

```
prompt> resolve_pg_nets -check_only
```

SEE ALSO

commit_upf(2)
connect_pg_net(2)

revert_blocks

Reverts one or more blocks of a sparse library to its version in the base library, removes the on-disk data of the design in the directory of the sparse library.

SYNTAX

return_val **revert_blocks**
blocks

Data Types

return_val The number of blocks reverted, 0 if none
blocks The single block name in
[libName:]blockName[/labelName][.viewName] format or
a collection of blocks

ARGUMENTS

blocks

A block name with optional library, label, and view specifications, or a collection of blocks. If the library specification is not present, the **current_lib** is used. If the label specification is not present, the default un-named label is used. If the view specification is not present, then *design view* is used. If the option is not given, the **current_block** is used. The blocks if opened must be removed/purged from the memory using **close_block** command before reverting them.

RETURN VALUE

Returns the number of blocks reverted, 0 if none. If an illegal name (with a slash) is given for the block, a TCL error is raised. If any specified block belongs to a non-sparse library, a TCL error is raised. If any specified block is open, a TCL error is raised.

DESCRIPTION

This command reverts a block and removes all of on-disk contents in the sparse library, retaining just the entry in the sparse library's catalog as remote. After it is executed the disk contents of the block are removed. Note that this command works only if the block is not loaded in memory.

EXAMPLES

This example reverts the block Top from catalog of the sparse library and removes it from disk.

```
prompt> revert_blocks Top
Information: Reverted design 'sparse_lib:Top.design' and removed it from disk. (NDMUI-365)
1
prompt> get_attribute [get_blocks -all Top] is_remote
true
```

SEE ALSO

close_block(2)
open_block(2)
copy_block(2)
move_block(2)
remove_blocks(2)
reopen_block(2)
open_lib(2)
get_blocks(2)
current_block(2)
get_designs(2)
current_design(2)

revert_cell_sizing

Reverts unacceptable ECO changes. This command is only for reverting ECO changes made by using the size_cell command that cause cell spacing rule violations.

SYNTAX

status **revert_cell_sizing**
-cells *cell_objects*
[-include_adjacent_sized_cells]
[-adjacent_cell_distance *distance*]

Data Types

cell_objects	string
distance	float

ARGUMENTS

-cells *cell_objects*

Specifies the cell objects on which the ECO changes are reverted. The cell objects can be a name string or collection.

-include_adjacent_sized_cells

Reverts adjacent sized cells, along with given sized cells, to avoid potential spacing rules violation. Adjacent cells are the cells on the same row that are close to the given cell. The search for adjacent cells is a recursive process, which means the tool reverts the adjacent sized cells of adjacent sized cells of the given sized cells until there is no adjacent sized cells.

-adjacent_cell_distance

Specifies the distance threshold for selecting the adjacent sized cells. The units are microns. The value cannot be negative. This option can only be used with the -include_adjacent_sized_cells option.

If the option is not used, the default value of the distance threshold between two cells is zero and the adjacent cells selected are the cells on the same row that abuts the given cell. If the option is used, adjacent cells selected are the cells on the same row that are within a distance that is less than or equal to the distance threshold.

DESCRIPTION

This command reverts unacceptable ECO changes on the specified objects. It rolls back the latest sizing changes applied on the objects.

The type of ECO changes supported by this command are those performed by the size_cell command.

When reverting the changes made by the size_cell command, the reference of cell is changed back to the original reference, before it was sized. Besides, the cell location and orientation of the cell are also reverted.

EXAMPLES

The following example reverts ECO changes by specifying the object name:

```
prompt> revert_cell_sizing cell_name
```

The following example reverts ECO changes by specifying a collection:

```
prompt> revert_cell_sizing [get_cells cell_name]
```

The following example reverts adjacent sized cells:

```
prompt> revert_cell_sizing [get_cells $revert_cells] \
    -include_adjacent_sized_cells
```

The following example reverts adjacent sized cells within a specified distance threshold:

```
prompt> revert_cell_sizing [get_cells $revert_cells] \
  -include_adjacent_sized_cells -adjacent_cell_distance 0.5
```

SEE ALSO

size_cell(2)

revert_eco_changes

Reverts unacceptable ECO changes. This command is for reverting ECO changes made by using size_cell, add_buffer and add_buffer_on_route.

SYNTAX

status **revert_eco_changes**
-cells *cell_objects*

Data Types

cell_objects collection

ARGUMENTS

-cells *cell_objects*

Specifies the cell objects on which the ECO changes are reverted. The cell objects can be a name string or collection.

DESCRIPTION

This command reverts unacceptable ECO changes on the specified objects.

The type of ECO changes supported by this command are those performed by the size_cell, add_buffer and add_buffer_on_route commands.

For newly added buffers, we can simply remove the buffers and recover the original connections. For inverters, we can identify inverter pairs and remove them at the same time to make sure the logic unchanged. For sized cell, the reference of cell is changed back to the original reference before it was sized. Besides, the cell location and orientation of the cell are also reverted.

EXAMPLES

The following example reverts ECO changes by specifying a collection:

prompt> **revert_eco_changes** -cells [get_cells $revert_cells]

SEE ALSO

size_cell(2)
add_buffer(2)
add_buffer_on_route(2)
report_eco_physical_changes(2)

rotate_objects

Rotates the specified objects.

SYNTAX

status **rotate_objects**
-orient *orient*
-angle *angle*
[-anchor center | ll | lr | ul | ur]
[-pivot {*x y*}]
[-group]
[-force]
[-simple]
[*object_list*]

Data Types

orient string
angle string
x float
y float
object_list collection or selection

ARGUMENTS

object_list

Collection or selection set that contains objects to rotate. If this option is not specified, global selection set is used.

-orient *orient*

Specifies the cell orientation.

The valid values are:

N, **W**, **S**, **E**,
FN, **FS**, **FE**, **FW**,
NW, **NE**, **EN**, **ES**,
SE, **SW**, **WN**, **WS**

-angle *angle*

Specifies the rotation angle of the objects.

The valid values are:

0, **90**, **180**, **270**,
CW90, **CW180**, **CW270**, **CCW90**, **CCW180**, **CCW270**
FLIPX, **FLIPY**

The rotation value definitions are:

CW90 is 90 degrees clockwise
CW180 and **CCW180** are 180 degrees
CW270 is 270 degrees clockwise
CCW90 is 90 degrees counterclockwise
CCW270 is 270 degrees counterclockwise
FLIPX is the reflection about the x-axis
FLIPY is the reflection about the y-axis

The following values are synonymous:

0, **R0**
CW90, **90**
CW180, **180**
CW270, **270**,
CCW90, **R90**
CCW180, **R180**
CCW270, **R270**,
FLIPX, **MY**
FLIPY, **MX**

-anchor center | ll | lr | ul | ur

Specifies anchor point for rotation.

The anchor values are:

center is center
ll is lower left
lr is lower right
ul is upper left
ur is upper right

The option cannot be specified together with the **-pivot** or **-group** option.

-pivot {*x y*}

Specifies the rotation pivot point.

The option cannot be specified together with the **-anchor** or **-group** option.

-group

Specifies the center point of a group as an anchor for rotation.

The option cannot be specified together with the **-anchor** or **-pivot** option.

-force

Resizes locked or fixed objects. By default, such objects are not resized according to global edit settings. Edit settings can be accessed using **get_edit_setting**.

-simple

Disable snapping and editing constraints. By default, objects are snapped according to global snap settings. Snap settings can be accessed using **get_snap_setting**.

DESCRIPTION

This command rotates one or more specified objects by given angle skipping all fixed objects. It can also set selected cell orientation to given value.

Snapping is done automatically using the global snap settings.

EXAMPLES

The following example rotates all selected objects by 90 degrees.

```
prompt> rotate_objects -angle 90
```

The alternative syntax uses a collection to specify objects.

```
prompt> rotate_objects [get_selection] -angle 90
```

SEE ALSO

change_selection(2)
copy_objects(2)
get_edit_setting(2)
get_selection(2)
get_snap_setting(2)
move_objects(2)
set_edit_setting(2)
set_snap_setting(2)
snap_objects(2)

route_3d_rdl

Routes 3DIC flip-chip nets.

SYNTAX

status **route_3d_rdl**
-layer *layer*
[-nets *collection_of_nets* | -nets_in_file *nets_file*]
[-objects *collection_of_objects*]
[-skip_detail_route true | false]
[-reuse_existing_global_route true | false]
[-coordinates *bbox_list*]

Data Types

layer collection
collection_of_nets collection
nets_file string
collection_of_objects collection
bbox_list list

ARGUMENTS

-layer *layer*

Specifies the target layer for 3DIC RDL routing by the names from the technology file.

-nets *collection_of_nets*

Specifies the flip-chip nets to route.

-nets_in_file *nets_file*

Specifies the name of the file that contains the list of RDL nets to route.

-nets has higher priority than **-nets_in_file**.

-objects *collection_of_objects*

Specifies the RDL nets (see **-nets**) by cells or pins. Only the sub-nets which involve the cells or pins are affected.

If the app option **flip_chip.route.routing_style** is **spanning_tree**, the whole nets of the cells or pins are regarded as involved.

If neither **-nets**, **-nets_in_file**, nor **-objects** option is specified, all flip-chip nets are routed.

-skip_detail_route *true* | *false*

Controls whether the command runs detail routing. If this option is true, the command skips detail routing and runs only global routing. By default, the option is false and the command runs both global routing and detail routing.

Only one of **-skip_detail_route** and **-reuse_existing_global_route** can be true.

-reuse_existing_global_route ***true*** | ***false***

Controls whether the command runs global routing. If this option is true, the command skips global routing and runs only detail routing by reusing the existing global route information. The default is false and the command deletes all existing global routes from the design before routing starts.

Only one of **-skip_detail_route** and **-reuse_existing_global_route** can be true.

-coordinates ***bbox_list***

Specifies the regions of routing for speeding up the routing process. If any input regions overlapped, then they are merged into a larger region. All the pins for each target net should be in the same routing region, and there needs to be enough space for routing inside the regions.

DESCRIPTION

This command routes 3DIC RDL nets using the RDL router. In addition, it can rip up and reroute wires.

A valid 3DIC RDL net is a net that connects a TSV to another TSV, a TSV to a bump cell, or a bump cell to another bump cell.

RDL routing is a special application where a shape-based router routes nets on specified layers. It meets the following requirements:

- Provides a routing tool to route, rip-up, and reroute automatically.
- Optimizes routing patterns to eliminate jogs and shorten wire length.
- Checks DRCs and opens for routed wires.
- Routes flip-chip nets on specified layers.

The suggested flow for RDL routing is as follows:

1. Define NDR rules by running the **create_routing_rule** command and then associate the rules to the flip-chip nets by the **set_routing_rule** command.
2. Set the RDL routing specific options by running the **set_app_options** command.

   Please specify flip_chip.route.point_to_point_connection_file_name and set flip_chip.route.design_tyle to 3dic_interposer for a high-bandwidth memory (HBM) design.
3. Create RDL I/O pad to bump assignments with the **create_matching_type** and **add_to_matching_type** commands if needed.
4. Perform RDL routing by running the **route_3d_rdl** command.
5. Manually route the open nets, if there are any, by using the **remove_routes**, **push_rdl_routes** and **route_3d_rdl** commands.
6. After all RDL nets are routed, improve the routing pattern by running the **optimize_rdl_routes** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example routes the flip-chip nets specified in the RDLNetFile.

```
prompt> route_3d_rdl -nets_in_file RDLNetFile -layer BM3
```

The following example routes all RDL nets.

```
prompt> route_3d_rdl -layer BM3
```

The following example places vias and routes in a HBM design.

```
prompt> create_interposer_routeplan
prompt> set_app_option -list {flip_chip.route.design_style 3d_interposer}
prompt> set_app_option -list {flip_chip.route.point_to_point_connection_file_name p2pfile.txt}
prompt> route_3d_rdl -layer BM3
```

SEE ALSO

add_to_matching_type(2)
create_matching_type(2)
route_rdl_flip_chip(2)
optimize_rdl_routes(2)
push_rdl_routes(2)
remove_routes(2)
create_interposer_routeplan(2)

route_auto

Performs global routing, track assignment, and detail routing in one step.

SYNTAX

status **route_auto**
[-max_detail_route_iterations *num*]
[-reuse_existing_global_route true | false]
[-route_nondefault_nets_first true | false]
[-stop_after_track_assignment true | false]
[-save_after_global_route true | false]
[-save_after_track_assignment true | false]
[-save_after_detail_route true | false]
[-save_cell_prefix *name*]

Data Types

num integer
name string

ARGUMENTS

-max_detail_route_iterations *num*

Specifies the maximum number of detail routing iterations. You can specify an integer between 1 and 1000.

By default, the maximum number of iterations for the detail routing command (40) is used.

-reuse_existing_global_route true | false

Enables or disables incremental global routing.

By default (false), the global router ignores all existing global routes from the design for initial routing.

If this option is set to true, the global router considers and reuses existing global routes.

-route_nondefault_nets_first true | false

Specifies whether to route the nets with nondefault routing rules before the nets that use the default routing rule.

When **false** (the default), nets with nondefault routing rules are not routed before the nets without nondefault routing rules.

When **true**, the tool performs global routing and track assignment first on nets with nondefault routing rules, and then on nets that use the default routing rule.

-stop_after_track_assignment true | false

Controls whether the command runs detail routing.

By default (false), the command runs global routing, track assignment, and detail routing.

If this option is true, the command skips detail routing and runs only global routing and track assignment.

-save_after_global_route true | **false**

Controls whether the design is saved after global routing.

The default value is false.

-save_after_track_assignment true | **false**

Controls whether the design is saved after track assignment.

The default value is false.

-save_after_detail_route true | **false**

Controls whether the design is saved after detail routing.

The default value is false.

-save_cell_prefix ***name***

Specifies the prefix to use when saving the design. For example, if you specify "foo" as the prefix, the name of the design saved after global routing is foo_icGRt.

The default prefix is auto.

DESCRIPTION

The **route_auto** command runs global routing, track assignment, and detail routing in one step. This command runs faster than using the basic commands because it does not access the disk between the different stages. The results are comparable to the basic flow (it might not be identical because the shapes could be ordered differently when the design is saved to disk).

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following command runs automatic routing using the **-save_after_global_route**, **-save_after_detail_route**, **-save_cell_prefix**, and **-max_detail_route_iterations** options:

```
prompt> route_auto \
   -save_after_global_route true \
   -save_after_detail_route true \
   -save_cell_prefix auto \
   -max_detail_route_iterations 15
```

SEE ALSO

route_global(2)
route_track(2)
route_detail(2)

route_busplans

Determines busplan topologies by using global routes.

SYNTAX

int **route_busplans**
[-force]
[-incremental]
[-quick [-reduce_virtual_pins]]
[-spread [-no_plan_busplans] [-skip_wns_check] [-reduce_virtual_pins]]
[*buses*]

Data Types

buses list of busplans

ARGUMENTS

-force

Normally the command includes a check that the horizontal and vertical layers are properly assigned to each bus by its associated busplan rule. Use this option to bypass this check to allow the router to run even if layers are not properly assigned. In this case, the router is free to use all layers to route. This option cannot be used with the **-quick** or **-reduce_virtual_pins** option. This option cannot be used with the **-spread** or **-no_plan_busplans** or **-skip_wns_check** options.

-incremental

This option specifies that all the planned buses that are not specified in the command are to have their busplans sent to router as a constraint. These planned buses are not going to be changed, but the influence of their busplans will be seen by router when routing the specified buses. A bus is considered to be planned if it has a valid net_estimation_rule and timing is met. In the busplan GUI, a planned busplan would be in green and not have any suggested registers in the layout window. This option cannot be used with the **-quick** option or **-reduce_virtual_pins** option. This option cannot be used with the **-spread** or **-no_plan_busplans** or **-skip_wns_check** options.

-quick

This option is labeled "Auto-plan" in the busplan GUI. It uses a modified greedy algorithm to quickly find a shortest path (where possible) for the specified busplans. If successful in planning a bus, virtual or real registers are spread, and if slack is negative, subsequently packed. This command is allowed to remove virtual pins not marked "fixed". Virtual pins marked "fixed" are considered as mandatory waypoints and will split a plan into 2 or more segments, each of which is planned individually. Where possible, each planned segment is rectilinear. This option cannot be used with the **-force** or **-incremental** options. This option cannot be used with the **-spread** or **-no_plan_busplans** or **-skip_wns_check** options.

Branching busplans (those with more than one endpoint per startpoint) are split into multiple separate paths and planned individually using this algorithm.

-reduce_virtual_pins

This option adds an intermediate reduction step to the **-quick** (Auto-plan) algorithm to try to create less virtual pins in the final plan. Note that the reduced paths produced by this option may not always be rectilinear. This option is for backward compatibility with previous auto-plan releases, and might be deprecated in the future. This cannot be used with the **-force** or **-incremental** options.

-spread

This option is labeled "Spread Plans" in the GUI. It uses a greedy, priority-based algorithm to allocate a portion of every channel to a particular busplan, where possible and no over-allocation exists. The list of busplans passed to this option is first adjusted to remove any branching busplans, or non-rectilinear busplans or busplans not meeting timing (attribute *timing_valid* is false, or attribute *worst_slack* is not at least slightly positive) or any busplans with a fixed element. If a busplan is too wide for a required channel, it is placed in the center of that channel. If a channel is over-subscribed, busplans will be placed on top of each other.

If a busplan is not rectilinear, an attempt to route/plan it using the **-quick** option (auto-plan) is first attempted, and if it still not rectilinear, it is discarded from the list of busplans to spread, unless the **-no_plan_busplans** option is specified.

The only busplans that are considered for channel spreading purposes are those passed in to one execution of this command - no context is saved between successive invocations. In general, all applicable busplans should be passed in at once, rather than one at a time. Passing in one busplan at a time will result in a much larger number of overlaps.

-no_plan_busplans

When used together with the *-spread* option, will not attempt to auto-plan busplans that are not rectilinear - instead, it will just skip them completely. This option can only be used together with *-spread*.

-skip_wns_check

When used together with the *-spread* option, will attempt to spread busplans that are rectilinear and having valid timing, but which may have negative slack. This option can only be used together with *-spread*.

buses

Specifies the busplans to route/plan. The buses must be created with the **create_busplans** command.

Application Options

This command is affected by the following application options, all of which are used ONLY by the **-quick** option. See the appropriate man page for details. **plan.busplan.channels_include_hm plan.busplan.channels_include_placement_blockage**

None of these options are used when the **-quick** option is not specified.

DESCRIPTION

This command determines the topology of busplans that are previously defined with the **create_busplans** command. The topology is determined by the global router (without **-quick**) or a greedy algorithm (with **-quick**).

EXAMPLES

The following example creates a busplan named bus1 and determines the topology for the busplan using global router.

```
prompt> set_net_estimation_rule fast_rule -parameter ns_per_mm -value 0.5
prompt> create_busplans -name bus1 -rule fast_rule -from B1/OUT1[*]
prompt> route_busplans bus1
```

This example re-plans the previous busplan using the Auto-plan option, first with virtual pin reduction, then with virtual pin reduction.

```
prompt> route_busplans -quick bus1
prompt> route_busplans -quick -reduce_virtual_pins bus1
```

SEE ALSO

create_busplans(2)
get_busplans(2)
modify_busplan(2)
set_net_estimation_rule(2)
write_busplans(2)
plan.busplan.channels_include_hm(3)
plan.busplan.channels_include_placement_blockage(3)

route_clock_straps

Performs routing on a group of clock nets that have clock straps. You should use this command before clock routing to route pins to clock meshes or other predefined physical clock structures.

SYNTAX

status **route_clock_straps**
-nets *nets*
[-topology comb|fishbone|sub_strap]
[-fishbone_fanout *fanout*]
[-fishbone_span *distance*]
[-fishbone_sub_span *distance*]
[-fishbone_layers *layer_names*]
[-sub_strap_layers *layer_names*]
[-sub_strap_max_delay *delay*]
[-stop_after_global_route true | false]
[-max_detail_route_iterations *num*]

Data Types

nets collection
fanout integer
distance integer
delay float
layer_names list
num integer

ARGUMENTS

-nets *nets*

Specifies the nets to be routed.

-topology comb|fishbone|sub_strap

Specifies the routing topology used to route each net. With comb routing, each pin is routed to a net shape with shape_use stripe, without sharing the connection with other pins, unless the nearest stripe is further away than the so-called comb distance. In case, there is no stripe shape within the comb distance, the connection would be similar to route_group behavior(includes steiner routing), in which case it may connect to non-stripe object as well create jogs for connection. By default, the comb distance is 2 time cell row height. The max comb distance that can be set is 10. However, larger the comb distance, longer is the routing runtime. Fishbone routing (the default) allows multiple load pins to share a so-called finger to connect to the nearest stripe; comb routing is used to connect the pins to the fingers. Each finger is a single straight shape that is routed orthogonally to the stripe. Driver pins get their own finger, which is not shared with other pins. In case of sub_strap routing, the sub-straps are created on same direction metal layer under pre-routes of type stripe. This leads to better routability as well as allows denser packing of the

pins under a substrap.

-fishbone_fanout ***fanout***

Specifies the target number of loads that get connected to a finger. A finger can connect to fewer loads if that is better for wirelength, it should not connect to more loads than the target fanout. The default value is 20.

-fishbone_span ***distance***

Specifies the maximum distance between any two loads connected to the finger, measured in the direction orthogonal to the direction of the finger. The actual span of a finger can be below the target if that is better for wirelength. By default a target span equal to 20 gcells distance is automatically derived.

-fishbone_sub_span ***distance***

Specifies the maximum distance between any two loads connected to the sub-finger, measured in the direction orthogonal to the direction of the sub-finger. The actual span of a sub-finger can be below the target if that is better for wirelength. The default value is 0, which means that no sub-fingers are created.

-fishbone_layers ***layer_names***

Specifies the routing layers that can be used for routing the fishbone fingers and sub-fingers. This option can only be used to reduce the allowed layer range, it cannot be used to extend the allowed routing layers for a net. By default, only the routing layers adjacent to the layer of the clock strap will be used.

-sub_strap_layers ***layer_names***

Specifies the routing layers that can be used for substraps. In case of multiple layers, layer parallel to the pre-route and closest to it will be picked for sub-strap creation. In case this option is not specified, the pre-route -2 layer is picked up for sub-strap creation.

-sub_strap_max_delay ***delay***

The delay from pre-route to sub-strap to the pins has to be contained to limit skew across sub-straps. In addition, this avoid overloading of Via Tower of sub-strap. Rough delay calculation would be done based on RC estimation of sub-strap and its allocated loads. This delay would be limited to 2 ps by default or a user specified value as the option specified.

-stop_after_global_route true | false

Controls whether the router stops after global routing or continues all the way to detail routing. By default, comb routing continues all the way to detail routing.

-max_detail_route_iterations ***num***

Specifies the maximum number of detail routing iterations for comb routing. You can specify an integer between 1 and 1000. By default, the maximum number of iterations is 40, the same number as for the detail routing command.

DESCRIPTION

The **route_clock_straps** command completes the routing of clock nets with clock straps, by routing the pins to clock straps with shape_use stripe. It supports two routing styles: the comb routing style and the fishbone routing style.

The **route_clock_straps** command honors the routing rule and the allowed routing layers defined on a net, including the settings defined with the **set_clock_routing_rules** command. The **-fishbone_layers** option can be used to explicitly define the layers that can be used for routing the fishbone fingers. It can only be used to reduce the set of allowed routing layers for a net. If more than one layer is available for routing a finger, the **route_clock_straps** command prefers the routing layers that are nearest to the clock strap that the finger connects to.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example uses fishbone routing to route pins to a clock mesh using a fishbone topology.

prompt> **route_clock_straps -nets [get_nets clk1_mesh]**

SEE ALSO

create_clock_straps(2)

route_custom

Performs custom routing with the Custom Router.

SYNTAX

status **route_custom**
  [-nets *nets*]
  [-keep_session true | false]

Data Types

nets collection

ARGUMENTS

-nets *nets*

Specifies the collection of nets to route. If no nets are specified, the command routes the entire design.

-keep_session true | false

Specifies whether or not the custom router view of the data is retained after the command completes. Specify *true* only if you expect to perform additional custom route commands immediately after the current command. The default is false.

DESCRIPTION

The **route_custom** command runs custom routing for a set of specified nets.

EXAMPLES

The following command routes the Data[0] net in the design using the Custom Router.

prompt> **route_custom -nets Data[0]**

SEE ALSO

route_detail

Performs detail routing on the block.

SYNTAX

status **route_detail**
[-max_number_iterations *count*]
[-coordinates *list_of_rectangles*
[-incremental true | false]
[-initial_drc_from_input true | false]
[-start_iteration *int*]

Data Types

count integer
list_of_rectangles list
int integer

ARGUMENTS

-max_number_iterations *count*

Specifies the maximum number of routing iterations. The detail router terminates after the maximum number of iterations or when it detects a lack of progress, whichever occurs first.

The range is from 1 to 1000. The default value is 40.

Limiting the number of routing iterations can provide faster results; however, it might cause the detail router to terminate before the design rule violations are resolved.

To disable automatic termination of detail routing (and force the router to complete the maximum number of iterations), set the **route.detail.force_max_number_iterations** application option to **true**.

-coordinates *list_of_rectangle*

Specifies the routing area in user units. You can specify multiple rectangular routing areas.

You must use the following syntax to specify the rectangles:

{ {{*llx1 lly1*} {*urx1 ury1*}} ...}"

If you do not specify this option, the entire chip is routed.

-incremental true | **false**

Controls whether the router uses incremental mode.

By default (**false**), the router does not use incremental mode.

When **true**, the router uses incremental mode and starts starts from the last iteration from the previous detail routing run on the block. To explicitly specify the starting iteration, use the **-start_iteration** option.

-initial_drc_from_input true | **false**

Controls whether the router uses the DRC information stored in the block.

By default, the command performs design rule checking on the entire block and then resolves the violations.

When **true**, the command skips design rule checking and uses the violations stored in the block as the starting point. Use this option very carefully; set it to **true** only if you are absolutely sure that the DRC information in the block are up-to-date.

-start_iteration ***int***

Specifies the start iteration for detail routing.

The router performs detail route iterations from the specified iteration up to the maximum iteration specified by the **-max_number_iterations** option.

The range is from 0 to 1000. The default is 0.

DESCRIPTION

This command performs detail routing on a block. You run this command after running the **route_track** command.

The behavior of this command is influenced by the route.detail and route.common application options.

Following is the treatment of shapes as fixed/unfixed within detail route.

Object NDM attribute value Router ---------- Net physical_status locked fixed wire/via user_route true fixed wire/via physical_status fixed/locked fixed wire/via physical_status firm fixed wire/via physical_status application_fixed not-fixed wire/via shape_use detail_route not-fixed wire/via shape_use shield_route not-fixed wire/via shape_use area_fill/active_fill not-fixed wire/via shape_use lib_cell_pin_connect not-fixed for detail route.

Object app option value Router ---------- Layer route.common.freeze_layer_by_layer_name layer-names fixed

When shape_use of a wire/via is lib_cell_pin_connect, check_routes treats the shape as not-fixed with the following exception:

- a net is frozen or a routing layer is frozen, AND - user runs check_routes with option check_from_frozen shapes and routing shape is on power/ground net. If the above conditions are satisfied, the shape is treated as fixed.

Route shapes with attributes not listed above will be treated as not-fixed.

Man page of shape_attributes describes supported values of "physical_status" for routing shapes. The settable attributes among them can be modified with command set_attribute or remove_attribute.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following command performs detail routing:

```
prompt> route_detail
```

The following command performs detail routing with a maximum of 3 iterations:

```
prompt> route_detail -max_number_iterations 3
```

The following command performs detail routing in the rectangle areas {0 0 50.0 60.2} and {100.0 100.0 120.2 110.2}:

```
prompt> route_detail \
  -coordinates { {{0 0} {50.0 60.2}} {{100.0 100.0} {120.2 110.2}} }
```

The following command performs incremental detail routing:

```
prompt> route_detail -incremental true
```

The following command uses the DRC information stored in the block and starts with iteration 1:

```
prompt> route_detail -initial_drc_from_input true \
  -start_iteration 1
```

SEE ALSO

route_auto(2)
route_global(2)
route_track(2)
route.common_options(3)
route.detail_options(3)

route_eco

Performs ECO routing on the design. This command should be used after signal routing has been completed. It should not be used to route nets before signal routing. Instead the group route command (**route_group**) should be used.

SYNTAX

status **route_eco**
[-max_detail_route_iterations *count*]
[-nets *list*]
[-cells *list*]
[-open_net_driven true | false]
[-max_reported_nets *limit*]
[-reroute modified_nets_only | modified_nets_first_then_others | any_nets]
[-reuse_existing_global_route true | false]
[-utilize_dangling_wires true | false]

Data Types

count integer
limit integer
list list

ARGUMENTS

-max_detail_route_iterations *count*

Specifies the maximum number of detail routing iterations. You can specify an integer between 1 and 1000.

By default, the maximum number of iterations for the detail routing command (40) is used.

-nets *list*

Specifies the nets on which to perform ECO routing. The command first connects the open nets in this list, and then fixes design rule violations within the bounding box of these nets.

By default, this command works on all nets in the design.

-cells *list*

Specifies the cells of the nets on which to perform ECO routing. The command fixes design rule violations within the bounding box of these cells.

By default, this command works on all cells in the design.

-open_net_driven true | false

Controls whether ECO route fixes DRC violations for the whole design (default) or only in the neighborhood (bounding box) of the open nets.

By default (false), ECO route connects the open nets in the design and fixes design rule violations for the entire design.

When true, ECO route connects the open nets in the design and fixes DRC violations only in the neighborhood of the open nets.

This option is ignored if the **-nets** option is specified.

-max_reported_nets ***limit***

Sets a limit on the number of changed nets to be reported at the end of the execution of this command. You can specify an integer between -1 and INT_MAX. When -1 is specified, the router will report all the nets that are changed.

By default, the limit is 100.

-reroute modified_nets_only | **modified_nets_first_then_others** | **any_nets**

Controls which nets are rerouted.

The definition for each mode is

* **any_nets** (default) The router reroutes any nets freely to fix the DRC violations. * **modified_nets_only** The router freezes all the fully connected nets and tries to finish the routing by modifying only the nets with open ECO changes. This can be very restrictive and can cause the router to fail to fix the violations. It is only suitable for very minor ECO changes. * **modified_nets_first_then_others** The router first freezes all the fully connected nets and tries to finish the routing by modifying only the nets with open ECO changes. If there are violations remaining, the router tries to reroute those fully connected nets to resolve the violations. This reduces the layout changes for minor ECO operations but is not suitable for major ECO changes.

-reuse_existing_global_route true | **false**

Enables or disables incremental global routing.

By default (false), the global router ignores all existing global routes from the design for initial routing.

If this option is set to true, the global router considers and reuses existing global routes.

-utilize_dangling_wires true | **false**

Controls whether the router tries to reuse the dangling routes as much as possible to fix the opens.

The default value is true.

DESCRIPTION

The **route_eco** command performs ECO routing. It first connects the open nets and then fixes the DRC violations. The options control which nets are connected and on which areas of the chip the DRCs are fixed.

This command should be used after signal routing has been completed. The group route command (**route_group**) should be used to route nets before signal routing.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following command performs ECO routing using the **-nets**, **-utilize_dangling_wires**, and **-reroute** options.

```
prompt> route_eco -nets {n1 n2 n3} -utilize_dangling_wires true \
   -reroute modified_nets_first_then_others
```

SEE ALSO

route_global(2)
route_track(2)
route_detail(2)
route.common(3)
route.detail(3)

route_fishbone

Performs fishbone style routing on a group of clock nets. Fishbone routing may not be shortest but reduces via count and clock skew.

SYNTAX

status **route_fishbone**
-nets *nets*
[-stop_after_global_route true | false]

Data Types

nets collection of nets

ARGUMENTS

-nets *nets*

Specifies the clock nets to be routed.

-stop_after_global_route true | false

Controls whether the router stops after global routing or continues all the way to detail routing. By default, the value is false and the router continues all the way to detail routing.

DESCRIPTION

The *route_fishbone* command completes the routing of clock nets with fishbone routing style. Fishbone routing is a structured routing topology where the net driver feeds a central fishbone trunk that spans the cluster of net loads being driven.

Several fishbone fingers can extend from the central trunk. The net loads connect directly to the fishbone fingers by way of comb routes, typically one comb route for each net load. The frequency of these fingers is controlled by the cts.routing.fishbone_max_tie_distance app option, which specifies a maximum distance from a net load to its nearest fishbone finger.

Optionally, you can use an additional level of fishbone routing, called sub-fingers. The sub-fingers connect to the fingers, and then the comb routes can instead connect the net loads to the sub-fingers. This can reduce wire length, especially in the case that a large fishbone_max_tie_distance is used. The behavior of sub-fingers is controlled by the cts.routing.fishbone_max_sub_tie_distance app option. The trunk, finger and sub-finger are defined as preroutes, using the shape_use==stripe attribute. After the route_fishbone command completes, the comb routes to pins are detail or global routed (depending on option "-stop_after_global_route"). The comb routes connecting the net loads would have shape_use==global_route or

shape_use==detail_route.

Fishbone trunks can optionally be biased toward power or ground nets to act as a shield on one side of the trunk, to improve signal integrity on the fishbone trunks. Biasing is enabled separately for horizontal and vertical trunks by setting a spacing rule between the trunk and the power or ground net using the cts.routing.fishbone_horizon- tal_bias_spacing and/or cts.routing.fishbone_vertical_bias_spacing app options.

When biasing is enabled, two additional controls are available. The cts.routing.fishbone_bias_threshold app option specifies a minimum trunk length before biasing is attempted. The cts.routing.fish- bone_bias_window app option specifies a maximum displacement from the preferred trunk location to the biased trunk location, above which biasing is not attempted.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example uses fishbone routing to route clock net using a fishbone topology.

prompt> **route_fishbone -nets [get_nets clk1]**

SEE ALSO

EXAMPLES

The following example uses fishbone routing to route clock nets using a fishbone topology.

prompt> **route_fishbone -nets [get_nets clk1]**

SEE ALSO

cts.routing.fishbone_max_tie_distance(3)
cts.routing.fishbone_max_sub_tie_distance(3)
cts.routing.fishbone_horizontal_bias_spacing(3)
cts.routing.fishbone_vertical_bias_spacing(3)
cts.routing.fishbone_bias_threshold(3)
cts.routing.fishbone_bias_window(3)
mark_clock_trees(2)
route_group(2)

route_global

Performs global routing on the design.

SYNTAX

status **route_global**
[-effort_level minimum | low | medium | high | ultra]
[-congestion_map_only true | false]
[-routing_utilization_only true | false]
[-reuse_existing_global_route true | false]
[-floorplan true | false]
[-virtual_flat all_routing | top_and_interface_routing_only]
[-host_options *host_option_name*]

Data Types

host_option_name string

ARGUMENTS

-effort_level minimum | **low** | **medium** | **high** | **ultra**

Specifies the effort level used to perform global routing. By default, the effort level is **medium**.

-congestion_map_only true | **false**

Creates the congestion map without creating global route segments (glinks). By default, this option is false and the command generates glinks. If design has existing glinks, the command will not remove them if this option is true and the new congestion map could potentially be out of sync comparing with the existing glinks.

-routing_utilization_only true | **false**

Creates the wire utilization maps without creating global route segments (glinks). Three wire utilization maps are output to indicate the track occupation by preroute, preroute plus clock nets, and all objects respectively. Extra memory consumption is needed to calculate and store wire utilization maps. By default, this option is false and the command generates glinks. If design has existing glinks, the command will not remove them if this option is true and the new wire utilization maps could potentially be out of sync comparing with the existing glinks.

-reuse_existing_global_route true | **false**

Specifies whether to reuse existing global routes. If this option is true, the global router considers and reuses existing global routes. By default, the option is false and the global router ignores all existing global routes for initial routing.

-floorplan true | **false**

Specifies whether to perform fast, lower effort global routing. If this option is true, the global router uses less effort to avoid

blockages on blocked pins and makes other simplifying assumptions to complete global routing quickly. Designs can be unlegalized in floorplan mode. Floorplan mode also disables timing or crosstalk-driven routing. The resulting congestion map shows the location of congestion hot spots in the floorplan. Absolute congestion statistics might differ from the non-floorplan mode global routing, especially if the design is not clean. Floorplan mode is intended to be used only in floorplan analysis or design planning to give a quick picture of floorplan routing problems as a guide for placement improvements. You should not run track assignment or detail routing based on floorplan mode global routing. By default, this option is false and the router routes the design with higher effort to reduce congestion.

-virtual_flat all_routing | top_and_interface_routing_only

Routes the design using a virtual flat approach while honoring all feedthrough or pin constraints and maintaining the physical hierarchy. In this mode, global router routes the top-level design and routes all the child-level blocks. This can be useful if the design has physical blocks and you want to route the whole design, including all child-level blocks.

When you specify **-virtual_flat all_routing**, the router routes all nets in both the top-level design and child-level blocks. When you specify **-virtual_flat top_and_interface_routing_only**, the router routes nets in top-level design and only the nets that connect to child-level blocks. Interface and feedthrough nets are routed crossing the block boundaries, and the resultant routes are stored in top-level and the corresponding child-level blocks.

After virtual flat global routing, you can place pins based on the global routing results with the **place_pins -use_existing_routing** command.

This option can only used together with **-floorplan true**.

-host_options *host_option_name*

Specifies that options set with the **set_host_options** command should be used for distributed processing. Note that this option can only be used together with the **-virtual_flat** option.

DESCRIPTION

The **route_global** command maps general pathways through the design for each unrouted signal or clock net. The global router uses a three-dimensional array of global routing cells to model the demand and capacity of global routing. The average height of the standard cells is used to create the height and width of each global routing cell. During global routing, the tool assigns nets to the global routing cells through which they pass. For each global routing cell, routing capacity is calculated according to the blockages, pins, and routing tracks inside the cell. Although global routing does not assign the nets to the actual wire tracks, it notes the number of nets assigned to each global routing cell. The tool calculates the demand for wire tracks in each global routing cell and reports the overflows, which are the number of wire tracks still needed after the tool assigns nets to the available wire tracks in a global routing cell. The tool might reduce overflows by detouring nets around congested areas and increasing the wire length. When a wire is prerouted, the tool checks whether it is completely connected. If the net is completely connected, the tool treats the net as a blockage. If the net is partially connected, the tool connects all parts of the net.

Note that the global router treats power and ground nets as blockages, and treats prerouted meshes and trunks as a single connection. Make sure that you examine the routing to determine whether the nets are connected properly.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following command performs global routing using high effort:

```
prompt> route_global -effort_level high
```

SEE ALSO

route_auto(2)

route_group

Performs routing on a group of nets in the design using router. You should use this command before signal routing to route critical nets such as clock nets or timing-critical nets. Do not use this command to connect nets after signal routing (use the **route_eco** command instead).

SYNTAX

status **route_group**
[-max_detail_route_iterations *num*]
[-nets *nets* [-from_file *filename*] | -all_clock_nets]
[-stop_after_global_route true | false]
[-reuse_existing_global_route true | false]
[-utilize_dangling_wires true | false]
[-floorplan true | false]
[-global_planning true | false]
[-route_nondefault_nets_first true | false]

Data Types

num integer
nets collection
filename string

ARGUMENTS

-max_detail_route_iterations *num*

Specifies the maximum number of detail routing iterations. You can specify an integer between 1 and 1000. By default, the maximum number of iterations is 40, the same number as for the detail routing command.

-nets *nets*

Specifies the nets to be routed. You must specify the nets on which to run this command by specifying at least one of the following options: **-nets**, **-all_clock_nets**, or **-from_file**. The **-nets** and **-all_clock_nets** options are mutually exclusive; you can specify only one. You can specify the **-from_file** option and the **-nets** option.

-from_file *filename*

Routes the nets specified by name in the file named *filename*. If the file does not exist or does not contain any valid nets, the tool generates an error message that indicates the source of the problem.

You must specify the nets on which to run this command by specifying at least one of the following options: **-nets**, **-all_clock_nets**, or **-from_file**. You can specify the **-from_file** option and the **-nets** option. However, the **-from_file** option cannot be used with the **-all_clock_nets** option.

-all_clock_nets

Routes all clock nets. You must specify the nets on which to run this command by specifying at least one of the following options: **-nets**, **-all_clock_nets**, or **-from_file**. If you specify the **-all_clock_nets** option, you cannot specify the **-nets** or **-from_file** option. Zroute's timing driven mode are turned off temporarily when running route_group -all_clock_nets.

-stop_after_global_route true | false

Controls whether the router stops after global routing or continues all the way to detail routing. By default, the value is false and the router continues all the way to detail routing. In this mode, we don't perform via ladder insertion. Please use refresh_via_ladders to update via ladders prior than route_group.

-reuse_existing_global_route true | false

Controls whether the tool performs incremental global routing. If you specify **-reuse_existing_global_route true**, the global router considers and reuses the existing global routes. By default, this option is false and the global router ignores all existing global routes from the design for initial routing.

-utilize_dangling_wires true | false

Controls whether the router tries to reuse the dangling routes as much as possible. By default, this option is true.

-floorplan true | false

Specifies the routing quality used for global routing.

If you specify **-floorplan true**, the global router uses less effort to avoid the blockages on blocked pins when routing the design and makes other simplifying assumptions to complete global routing quickly. Designs can be unlegalized in floorplan mode. Floorplan mode also disables timing and crosstalk-driven routing. The resulting congestion map shows the location of congestion hot spots in the floorplan. Absolute congestion statistics might differ from the normal global router, especially if the design is not clean. Floorplan mode should be used only in floorplan analysis or design planning. Its purpose is to give a quick picture of floorplan routing problems as a guide for placement improvements. You should not run track assignment or detail routing based on floorplan mode global routing. If you omit the **-stop_after_global_route true** option when you specify **-floorplan true**, the router sets **-stop_after_global_route true** and issues an error message indicating that floorplan mode global routing should not be used for detail routing.

By default, this option is false and the router routes the design in normal mode using higher effort to reduce congestion.

-global_planning true | false

Controls whether the global router routes the design in a virtual flat way while honoring all the feedthrough or pin constraints with physical hierarchy awareness. In this mode, global router connects to pins inside child blocks when routing a net. This is useful when you want to route the top-level net while considering the objects inside the child blocks. The router honors routing blockages, route guide, hardmacros and pre-route inside the child blocks.

If this option is used together with the **-stop_after_global_route true** option, the router outputs global route results and skips wire track assignment and detail routing. In addition, when the net is connected to a child block, the router creates pins for the child block ports that connected to this net. The pins have the same shape, layer, and location as the pins inside child block that connect to the same top-level net. After global routing, you can use the **place_pins -use_existing_routing** command to create feedthroughs and place pins based on this global route result. This method will only create feedthroughs and pins on the blocks instantiated in the next level below, not at all levels of hierarchy.

-route_nondefault_nets_first true | false

Specifies whether to route the nets with nondefault routing rules before the nets that use the default routing rule.

When **false** (the default), nets with nondefault routing rules are not routed before the nets without nondefault routing rules.

When **true**, the tool performs global routing and track assignment first on nets with nondefault routing rules, and then on nets that use the default routing rule.

DESCRIPTION

The **route_group** command routes a specified group of nets. You should use this command before signal routing to route critical nets such as clock nets or timing-critical nets. After signal routing, do not use this command to connect nets; use the **route_eco** command instead.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following command performs routing for all clock nets using the **-max_detail_route_iterations** and **-utilize_dangling_wires** options.

```
prompt> route_group -all_clock_nets \
  -max_detail_route_iterations 6 \
  -utilize_dangling_wires true
```

SEE ALSO

route_detail(2)
route_global(2)
route_track(2)
route_eco(2)

route_opt

Optimize the current routed design.

SYNTAX

route_opt

DESCRIPTION

This command optimizes the current routed design for timing, electrical DRC violations, area, and power. It performs legalization and ECO routing on the optimized design. By default, power optimization, concurrent clock and data optimization, and crosstalk reduction are not performed by this command. Use application options in the "SEE ALSO" section to enable them.

SEE ALSO

route_auto(2)
route_opt.eco_route.mode(3)
route_opt.flow.enable_ccd(3)
route_opt.flow.enable_power(3)

route_rdl_differential

Routes RDL nets and matches the route length or the pattern for each net.

SYNTAX

status **route_rdl_differential**
-layers *layer_list*
[-nets *collection_of_nets* | -nets_in_file *nets_file*]
[-objects *collection_of_objects*]
[-mode length | parallel]
[-allow_push_neighbor_nets true | false]

Data Types

layer_list collection
collection_of_nets collection
nets_file string
collection_of_objects collection

ARGUMENTS

-layers *layer_list*

Specifies the target layers for RDL differential routing from the technology file. The number of layers is limited to no more than 2 and the layers must be adjacent to each other. The command considers the first layer in the list as the primary layer.

-nets *collection_of_nets*

Specifies the RDL nets on which to perform differential routing. The number of nets must be 16 or fewer.

-nets_in_file *nets_file*

Specifies the name of the file that contains the list of RDL nets on which to perform differential routing. The number of nets must be 16 or fewer.

-nets has higher priority than **-nets_in_file**.

-objects *collection_of_objects*

Specifies the RDL nets (see **-nets**) by cells or pins. Only the sub-nets which involve the cells or pins are affected.

If the app option **flip_chip.route.routing_style** is **spanning_tree**, the whole nets of the cells or pins are regarded as involved.

-mode length | parallel

Specifies the mode of differential routing. In length mode, the router routes the RDL nets and matches the route length. In parallel

mode, the router routes the RDL nets in parallel in the middle part and routes separately to each pins. Thus, in this mode, the routing pattern will match in the middle part of the routing.

The default value is length.

-allow_push_neighbor_nets true | false

Specifies if the router can push away the neighbors of the target nets in order to achieve better length matching quality. The default value is false.

This option applies to length matching mode only.

DESCRIPTION

This command routes the specified nets on the redistribution layers and matches the route lengths. After running this command, do not use the **route_rdl_flip_chip**, **push_rdl_routes**, or **optimize_rdl_routes** command to further process the group of nets.

RDL nets are nets in flip-chip designs which are routed on the redistribution layer. In a flip-chip design, RDL nets connect a bump cell to driver I/O cells.

The suggested flow for RDL differential routing is as follows:

1. Run the **route_rdl_differential** command on the group of target nets to perform RDL differential routing.
2. Check the report generated by **route_rdl_differential** and adjust the route length or pattern manually if needed.
3. Perform RDL routing on other nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example routes the RDL nets specified in the RDLNetFile and matches the RDL route length.

```
prompt> route_rdl_differential -nets_in_file RDLNetFile -layers M9
```

The following example perform the differential routing parallel mode.

```
  .in +0.25in
prompt> route_rdl_differential -nets {NetA NetB} -mode parallel -layers M9
```

SEE ALSO

route_rdl_flip_chip(2)

route_rdl_flip_chip

Routes flip-chip nets.

SYNTAX

status **route_rdl_flip_chip**
-layers *layer_list*
[-nets *collection_of_nets* | -nets_in_file *nets_file*]
[-objects *collection_of_objects*]
[-skip_detail_route true | false]
[-reuse_existing_global_route true | false]
[-coordinates *bbox_list*]

Data Types

layer_list collection
collection_of_nets collection
nets_file string
collection_of_objects collection
bbox_list list

ARGUMENTS

-layers *layer_list*

Specifies the target layers for flip-chip routing by the names from the technology file. Currently, the number of layers is limited to 2, and the layers must be adjacent to each other. The command treats the first layer as the primary layer.

-nets *collection_of_nets*

Specifies the flip-chip nets to route.

-nets_in_file *nets_file*

Specifies the name of the file that contains the list of flip-chip nets to route.

-nets has higher priority than **-nets_in_file**.

-objects *collection_of_objects*

Specifies the RDL nets (see **-nets**) by cells or pins. Only the sub-nets which involve the cells or pins are affected.

If the app option **flip_chip.route.routing_style** is **spanning_tree**, the whole nets of the cells or pins are regarded as involved.

If neither **-nets**, **-nets_in_file**, nor **-objects** option is specified, all flip-chip nets are routed.

-skip_detail_route ***true*** | ***false***

Controls whether the command runs detail routing. If this option is true, the command skips detail routing and runs only global routing. By default, the option is false and the command runs both global routing and detail routing.

Only one of **-skip_detail_route** and **-reuse_existing_global_route** can be true.

-reuse_existing_global_route ***true*** | ***false***

Controls whether the command runs global routing. If this option is true, the command skips global routing and runs only detail routing by reusing the existing global route information. The default is false and the command deletes all existing global routes from the design before routing starts.

Only one of **-skip_detail_route** and **-reuse_existing_global_route** can be true.

-coordinates ***bbox_list***

Specifies the regions of routing for speeding up the routing process. The regions are applied to all RDL layers. If any input regions overlapped, then they are merged into a larger region. All the pins for each target net should be in the same routing region, and there needs to be enough space for routing inside the regions.

DESCRIPTION

This command routes flip-chip nets using the flip-chip router. In addition, it can rip up and reroute wires.

A flip-chip net is a net that connects a flip-chip driver I/O cell to a bump cell, which is also called a flip-chip pad.

Flip-chip routing is a special application where a shape-based router routes nets on specified layers. It meets the following requirements:

- Provides a routing tool to route, rip-up, and reroute automatically.
- Optimizes routing patterns to eliminate jogs and shorten wire length.
- Checks DRCs and opens for routed wires.
- Routes flip-chip nets on specified layers.

The suggested flow for flip-chip routing is as follows:

1. Define NDR rules by running the **create_routing_rule** command and then associate the rules to the flip-chip nets by the **set_routing_rule** command.
2. Set the flip-chip routing specific options by running the **set_app_options** command.
3. Create flip-chip I/O pad to bump assignments with the **create_matching_type** and **add_to_matching_type** commands if needed.
4. Perform flip-chip routing by running the **route_rdl_flip_chip** command.
5. Manually route the open nets, if there are any, by using the **remove_routes**, **push_rdl_routes** and **route_rdl_flip_chip** commands.
6. After all flip-chip nets are routed, improve the routing pattern by running the **optimize_rdl_routes** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example routes the flip-chip nets specified in the RDLNetFile.

```
prompt> route_rdl_flip_chip -nets_in_file RDLNetFile -layers M9
```

The following example routes all flip-chip nets.

```
prompt> route_rdl_flip_chip -layers M9
```

SEE ALSO

add_to_matching_type(2)
create_matching_type(2)
optimize_rdl_routes(2)
push_rdl_routes(2)
remove_routes(2)

route_track

Performs track assignment on the design.

SYNTAX

status **route_track**

ARGUMENTS

This command has no arguments.

DESCRIPTION

This command performs track assignment on a global routed design. Run this command after running **route_global** and before running **route_detail** to assign wires to the routing tracks defined in the technology file. Track assignment operates on the entire design at once; whereas the detail router routes a small area at a time. Track assignment can make long routes straight and reduce the number of vias. After track assignment finishes, all nets are routed, but there are many violations, particularly where the routing connects to pins. The detail router fixes the violations.

The behavior of this command is influenced by options **route.track** and **route.common** set by the **set_app_options** command.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following command performs track assignment.

```
prompt> route_track
```

SEE ALSO

route.track(3)
route.common(3)
route_auto(2)
route_global(2)
route_detail(2)
route_group(2)
route_eco(2)

run_block_compile_pg

Runs distributed power and ground (PG) creation for the top-level and block-level designs. This command creates power and ground within the blocks for the block-level designs.

SYNTAX

status **run_block_compile_pg**
[-skip_top]
[-host_options *host_options_name*]

Data Types

host_options_name string

ARGUMENTS

-skip_top

Do not create PG network in top-level design.

-host_options *host_options_name*

Uses the specified host option for distributed processing. If this option is not specified, the PG network is created sequentially, beginning with the top-level design and followed by each reference block.

DESCRIPTION

This command runs distributed PG creation for the top-level and block-level designs.

Before running this command, you must define the full-design PG constraints, including PG via master rules, PG patterns, PG strategies, and PG strategy via rules. A separate **compile_pg** script, which contains any application options and **compile_pg** commands, is also required. You must run the **characterize_block_pg** command first to characterize the PG constraints for each block, as well as the top-level design. The top-level PG is created based on the currently defined PG constraints and the specified **compile_pg** command script while for each block the PG is created based on the characterized PG constraints and the **compile_pg** command script. The *pg_mapfile* PG mapping file is created from the characterization. The mapping file must be set with the **set_constraint_mapping_file** command, before using the **run_block_compile_pg** command for distributed PG creation.

Not all PG related commands are supported. Currently **check_pg_missing_vias**, **check_pg_connectivity**, and **check_pg_drc** are not supported for distributed PG creation.

EXAMPLES

The following example runs PG characterization, sets the mapping file, and starts distributed PG creation for up to the first-level of physical hierarchy.

```
prompt> set_host_options -name my_hosts -submit_command "sh"
prompt> set_editability -value false -from_level 2
prompt> characterize_block_pg -compile_pg_script compile_pg.tcl
prompt> set_constraint_mapping_file ./pgroute_output/pg_mapfile
prompt> run_block_compile_pg -host_options my_hosts
```

The following example runs PG characterization, sets the mapping file, and starts distributed PG creation for all levels of physical hierarchy.

```
prompt> set_host_options -name my_hosts -submit_command "sh"
prompt> characterize_block_pg -compile_pg_script compile_pg.tcl
prompt> set_constraint_mapping_file ./pgroute_output/pg_mapfile
prompt> run_block_compile_pg -host_options my_hosts
```

The following example runs PG characterization, sets the mapping file, and starts distributed PG creation for up to the second level of physical hierarchy.

```
prompt> set_host_options -name my_hosts -submit_command "sh"
prompt> set_editability -value false -from_level 3
prompt> characterize_block_pg -compile_pg_script compile_pg.tcl
prompt> set_constraint_mapping_file ./pgroute_output/pg_mapfile
prompt> run_block_compile_pg -host_options my_hosts
```

The following example runs PG characterization, sets the mapping file, and starts distributed PG creation for all levels of physical hierarchy excluding the AUINPUT block.

```
prompt> set_host_options -name my_hosts -submit_command "sh"
prompt> set_editability -value false -blocks AUINPUT
prompt> characterize_block_pg -compile_pg_script compile_pg.tcl
prompt> set_constraint_mapping_file ./pgroute_output/pg_mapfile
prompt> run_block_compile_pg -host_options my_hosts
```

SEE ALSO

characterize_block_pg(2)
compile_pg(2)
set_constraint_mapping_file(2)
set_editability(2)
set_host_options(2)

run_block_script

Runs the specified Tcl script on a set of blocks.

SYNTAX

status **run_block_script**
-script *script_name*
-cells *collection_of_cells*
-blocks *list_of_blocks*
-run_order bottom_up | top_down
[-host_options *host_option_name*]
[-var_list *list_of_variable_value_pairs*]
[-reuse_processes]
[-work_dir *directory*]
[-force]
[-name *job_name*]

Data Types

script_name	string
collection_of_cells	collection of cells
list_of_blocks	list of block names
host_option_name	name of a host option
list_of_variable_value_pairs	list of variables and their values
directory	working directory
job_name	string

ARGUMENTS

-script *script_name*

Specifies the script to run for each specified block.

-cells *collection_of_cells*

Specifies the cells to run. Each cell specified in the list must be a block. If you specify cells with the same master (if they are Multiply Instantiated Blocks), then only one will be run. The **-blocks** option is mutually exclusive with the **-cells** option.

-blocks *list_of_blocks*

Specifies the names of the blocks to run. The **-blocks** option is mutually exclusive with the **-cells** option.

-run_order bottom_up | top_down

Specifies the order to run the blocks. Valid values are *top_down* or *bottom_up*. The default is *bottom_up*.

In a multilevel physical hierarchy design, blocks have to run in a certain order. Jobs for any pair of blocks with ancestor/descendant relation cannot be run in parallel, which means the blocks need to run in either top-down order or bottom-up order. Use this option to control which order to use.

With top-down order, a block cannot start to run until all the jobs for the blocks that this block is instantiated in finishes. With bottom-up order, a block cannot start to run until all its child blocks finishes.

-host_options *host_option_name*

Specifies the name of the host option to use for distributed processing. Host options are created using the **set_host_options** command. Before using this option, you should test the *host_option_name* argument with the **check_host_options** command.

-var_list *list_of_variable_value_pairs*

Specifies the Tcl variables and settings to pass to the block runs. For example, to set variable a to 1 and variable b to 2 and pass the settings to the block runs, use one of the following formats for the option arguments:

{{a} {1} {b} {2}}
"a 1 b 2"

To pass variable c with value $c, use one of the following formats for the option arguments:

[list [list c] [list $c]]
"c $c"

-reuse_processes

Specifies that more than one block can be run in the same process. By default, each block is run in a separate process during a parallel run. That is because it ensures that each block run is independent of other blocks. However sometime there is a high cost to launching new processes. In those cases, you can use this option to enable running more than one block in the same process. However in this case, you need to take care that the input script is clean. See description section for an example of an issue that could arise.

-work_dir *directory*

Specifies a directory in which to write data and log files from the run. By default, the tool uses the ./work_dir directory. If you also specify the **-host_options** option, the **-work_dir** directory must be a network disk that is accessible to all clients.

-force

Forces the command to run, even if some blocks might be dirty. The distributed run proceeds and any unsaved modifications are lost. By default, the command issues an error message if any block is modified but not saved, and gives you a chance to save the changes.

-name *job_name*

Gives a prefix name to the job to run, rather than the system default prefix name. This allows for jobs to be labeled with a name meaningful to the task.

DESCRIPTION

This command executes a Tcl script on a specified collection of blocks. The command sets five Tcl variables while running the blocks: *block_refname*, *block_refname_no_label*, *block_libfilename*, *top_libname*, and *work_dir*. You can use these variables in the Tcl script specified by the **-script** option. *block_refname* is the reference name of the block being processed including the block label. *block_refname_no_label* is the reference name of the block being processed without the block label. *block_libfilename* is the full path name of the library of the block being processed. *top_libname* is the name of the library for the top-level design which contains the blocks being processed. *work_dir* is a subdirectory named *block_refname* in the directory specified by the **-work_dir**

option.

If the **-host_options** option is specified, the blocks are run in parallel, otherwise they are run one-by-one. Before using this option, you should check the host options with the **check_host_options** command. Note that the **run_block_script** command with the **-host_options** option closes all reference libraries, so all reference libs must be clean for the command to work properly. You should save all reference libraries with the **save_lib -all** command, but do not close the reference libs.

If you use the **-reuse_processes** option, or if you run sequentially, then you need to be careful when creating the block script. In particular, you must create the script to run in an order-independent fashion to generate consistent results. One common issue is setting a variable for a specific block A, but not clearing the variable after block A completes. The script might appear to work, but if another block is run after block A on the same process, the new block inherits the special variable setting. This might cause a problem or an unwanted side effect. Note that if you set a variable in the block script and run serially, the master process inherits the variable setting.

APP OPTIONS

The following app options can be used to control the behavior of run_block_script: plan.distributed_run.block_priority(3) plan.distributed_run.priority_dependencies(3) plan.distributed_run.priority_size(3) plan.distributed_run.reuse_rbs_workers(3)

EXAMPLES

The following example block script and command illustrate the basic use of the **run_block_script** command.

```
# Block script run.tcl
open_lib $block_libfilename
open_block $block_refname
echo "run" $block_refname
close_block $block_refname
close_lib $block_libfilename

prompt> set_host_option -name myhost -submit_command "bsub"
prompt> run_block_script -script run.tcl -cells {cell1 cell2} -host_options myhost
Distributed run start.
Worker_msg: block block1 log : <path to log for block1>
Worker_msg: block block2 log : <path to log for block2>
Block block1
  Wall clock time spent pending : 00:00:04.06
  Wall clock time spent running : 00:00:08.06
Block block2
  Wall clock time spent pending : 00:00:08.11
  Wall clock time spent running : 00:00:07.03
Distributed run summary
  Wall clock time (total)            : 00:00:45.20
  Longest running block              : block1
  Wall clock time lost due to pending : 00:00:08.08
Distributed run end.
...
prompt>
```

SEE ALSO

check_host_options(2)
report_host_options(2)
save_lib(2)
set_host_options(2)

run_monitor_gui

Starts the Synopsys Distributed Processing Monitor GUI program.

SYNTAX

pid **run_monitor_gui**
[-kill]
[-nolaunch]

Data Types

pid process id of started program, 0 on failure.

ARGUMENTS

-kill

Terminates (kills) any existing monitor instances before launching.

-nolaunch

Prevents the command from starting a new distributed process manager window. Use the **-nolaunch** and **-kill** options together to close all running instances of the Distributed Processing Monitor GUI, without starting a new instance.

DESCRIPTION

This command launches one instance of the Synopsys Distributed Processing Monitor GUI program, up to a maximum of two instances. The Monitor GUI can be used to watch the progress of all distributed tasks, such as the **run_block_script** or **create_abstract**, if the host options are set to use distributed processing with the **set_host_options** command. The distributed workers periodically send updates to the Monitor GUI to show the currently running command, RAM and CPU usage. Not all commands update the Monitor GUI in this way. The command returns the process id for the dpmanager task that was created.

If there are any other Monitor GUI tasks still running when the current session ends, they are also terminated before the session exits.

There is a limit of two running instances at the same time. The IC Compiler II shell tracks the number of running instances and enforces this limit. If there are existing Monitor GUI instances from another Synopsys product that are using the CDPL Distributed Library, or a previous session, they can also connect to the current session to act as a monitor. In this case, even though the shell will launch two instances, one or both of the instances might not connect to the current session and display worker progress. If the instance cannot connect, close the existing Monitor GUI instances or use the other sessions instead.

The Monitor GUI can also be started independent of a session by running **dpmanager -n** as a UNIX/Linux command from the

cdpl/bin directory of the distribution image.

EXAMPLES

The following example starts a new instance of the Monitor GUI if less than two instances already running. The command returns the process number for the dpmanager task started by **run_monitor_gui**.

```
prompt> run_monitor_gui
15123
```

The following example start a new instance of the Monitor GUI, but first terminates all running Monitor GUI instances launched from this session. The command returns the process number for the dpmanager task started by **run_monitor_gui**.

```
prompt> run_monitor_gui -kill
15254
```

The following example terminates all running Monitor GUI instances launched from this session.

```
prompt> run_monitor_gui -kill -nolaunch
```

SEE ALSO

create_abstract(2)
create_placement(2)
estimate_timing(2)
merge_abstract(2)
run_block_compile_pg(2)
run_block_script(2)
set_host_options(2)
shape_blocks(2)

run_test_point_analysis

Runs SpyGlass DFT testability analysis.

SYNTAX

status **run_test_point_analysis**

ARGUMENTS

This command has no arguments.

DESCRIPTION

The **run_test_point_analysis** command performs SpyGlass DFT analysis to determine the test points to be inserted by the **testability** DFT client.

Testability analysis is configured by the **set_testability_configuration** command. For details, see that man page.

This command is supported only in the gate-level flow.

Before running this command, the SPYGLASS_HOME environmental variable (not Tcl application variable) must be set so the tool can find the **spyglass** binary. You can check this as follows:

```
prompt> echo $env(SPYGLASS_HOME)
/global/apps/spyglass_2016.06-SP2
```

When you run this command, it reports information about the SpyGlass DFT analysis as it runs, such as coverage values and number of test points found.

The following testability targets do not require you to run testability analysis (although doing so is harmless):

- The **core_wrapper** target
- The **user** target

EXAMPLES

The following example configures and runs test point analysis:

```
# enable and configure testability analysis
set_dft_configuration -testability enable
set_testability_configuration -target {random_resistant}
set_testability_configuration -target {untestable_logic}

# perform pre-DFT DRC
create_test_protocol

# perform testability analysis
run_test_point_analysis
```

SEE ALSO

set_dft_configuration(2)
set_testability_configuration(2)

run_vclp_cmd

Used to execute a command in VCLP from inside ICC2/Fusion Compiler shell.

SYNTAX

status **run_vclp_cmd "command name"**

DESCRIPTION

The **run_vclp_cmd** command is used to execute a command in VCLP from ICC2/FC Shell. The command to be run in VCLP must be specified inside double quotes.

Example: run_vclp_cmd "report_lp"

SEE ALSO

check_mv_design(2)

saif_map

Controls the name-mapping database for pins, ports, cells and nets through its various options.

SYNTAX

status **saif_map**
[-start]
[-stop]
[-reset]
[-report]
[-get_saif_names *object_list*]
[-write_map *file_name* [-type ptpx | primepower [-saif_file *file_name* [-strip_path *source_name*]] [-essential]]]
[-read_map *file_name*]
[-set_name *object_list*]
[-add_name *object_list*]
[-inverted]
[-remove_name *object_list*]
[-get_object_names *saif_name_list*]
[-change_name *{saif_name changed_name}*]

Data Types

object_list	collection
file_name	string
source_name	string
saif_name_list	string list
saif_name changed_name	string pair

ARGUMENTS

-start

Starts the internal name-mapping mechanism. The name transformations are stored and are tracked, only after the command is run with this option. Reading and writing operations can also be performed if the command has been run with this option.

-stop

Stops the internal name-mapping mechanism. The name transformations stored are deleted. To begin tracking of the name transformations, the **saif_map** command has to be restarted. After the command has been run with this option, no **saif_map** feature works, unless the **saif_map** is restarted.

-reset

Deletes the manual SAIF name mapping from the database.

-report

Reports any manual SAIF name mapping you set.

-get_saif_names *object_list*

Reports the SAIF names corresponding to the specified objects. An *object_list* must be provided when this option is specified.

-write_map *file_name* -type ptpx | primepower -saif_file *file_name* -strip_path *source_name* -essential

Writes the name-mapping information into the file specified by *file_name*. The **-type ptpx** or **-type primepower** writes the name-mapping information in PrimeTime PX file format. When the optional **-saif_file** option is given with PrimeTime PX file format, only mappings for the names that occur in the specified SAIF file are written out. If no **-saif_file** option is given, PrimeTime PX mappings for all objects which have changed from original netlist (the netlist when **saif_map -start** was executed) are written out. The **-strip_path** option can be specified only with **-saif_file**. **-strip_path** specifies the name of the instance of the current design as it appears in SAIF. It is equivalent to the **-strip_path** option of the **read_saif** command. The **-essential** option writes only the essential points in the PrimeTime PX mapping file. Option **-essential** and **-saif_file** are mutually exclusive. Also option **-essential** can only be specified with **-type ptpx** or **-type primepower**.

-read_map *file_name*

Reads the name-mapping information from the file specified by **file_name**. The name-mapping information read from this file is written over the existing name-mapping information stored internally.

-set_name *object_list*

Sets the given SAIF name for specified objects in the name-mapping database, replacing any existing mapping for the objects. If a pin or a port is specified, the mapping is applied to the net connected to the pin/port. The **-inverted** option can be specified along with this option, to indicate that the SAIF name maps to a logically inverted object. An *object_list* must be provided when this option is specified.

-add_name *object_list*

Adds the given SAIF name for specified objects in the name-mapping database. The **-inverted** option can be specified along with this option, to indicate that the SAIF name maps to a logically inverted object. An *object_list* must be provided when this option is specified.

-inverted

Specifies that the name to be set maps to a logically inverted object. This option is valid only with the **-set_name** and **-add_name** options. For example, you can use the **-inverted** option when specifying that the object A_reg/QN maps to the SAIF name A, but its logical value is inverted.

-remove_name *object_list*

Removes the given SAIF name for specified objects in the name-mapping database. An *object_list* must be provided when this option is specified.

-get_object_names *saif_name_list*

Returns the objects to which the specified SAIF names map in the name-mapping database.

-change_name {*saif_name changed_name*}

Interprets the *saif_name* as the *changed_name* while parsing the SAIF file.

DESCRIPTION

The **saif_map** command controls the name transformations' tracking, storage, writing and reading. The objects that are tracked through this command are - pins, ports, nets, and cells. This command provides several options for reporting, querying, and modifying the SAIF name-mapping database information. To use any saif_map option, you need to have first specified **saif_map -start**, or **saif_map -read_map**, which starts the name mapping mechanism. You can write the name-mapping database information to a file by using the **-write_map** option. You can read the information back into the design by using the **-read_map** option. Using the **-set_name**, **-add_name**, and **-remove_name** options, you can manually set your name mappings for specific objects. Note that if the object specified is a pin or a port, the mapping is applied on the net connected to the pin or port. The **-inverted** option can be specified along with the **-set_name** and **-add_name** options, to indicate that the object corresponding to the specified name is logically inverted. You can change the way a particular SAIF name is interpreted while parsing the SAIF file by using the **-change_name** option. Note that both the *saif_name* and the *changed_name* need to be valid hierarchical names for the interpretation to be successful. To report the manual name mappings, you can use the **-report** option, which generates a report in the Tcl command format. The **-reset** option clears any user-specified manual name mappings. The **-get_saif_names** and **-get_object_names** options can be used to see the mappings for certain objects and SAIF names, respectively. The output of the **-get_saif_names** option consists of the *object_type*, the *object_name*, a list of SAIF names which can be empty, and optionally, a list of inverted SAIF names, as shown as follows:

{{*object_type object_name*} {saif_names {*saif_names*}}
  {inverted_saif_names {*inverted_saif_names*}}}

The return value of the -get_object_names option is a Tcl list with an element for each specified SAIF name. The format is:

{*saif_name* {objects {*objects*}} {inverted_objects {*inverted_objects*}}}

Each object or inverted object is formatted as {object_type object_name}.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example starts the name transformation tracking.

```
prompt> saif_map -start
```

The following example stops the name transformation tracking, and deletes the internally stored name-mapping information.

```
prompt> saif_map -stop
```

The following example writes the name-mapping information using the default hierarchy separator in the test1.map file.

```
prompt> saif_map -write_map test1.map
```

The following example writes the name-mapping information in PrimeTime PX command format in the test1.ptpx file. Only name mappings for names that occur in SAIF file test.saif are written. This results in a SAIF-file specific SAIF map.

```
prompt> saif_map -write_map test1.ptpx -type ptpx -saif_file test.saif
```

The following example writes the name-mapping information in PrimePower command format in the test1.primepower file. Only name mappings for names that occur in SAIF file test.saif are written. This results in a SAIF-file specific SAIF map.

```
prompt> saif_map -write_map test1.primepower -type primepower -saif_file test.saif
```

The following example reads the name-mapping information from the test1.map file.

```
prompt> saif_map -read_map test1.map
```

The following example sets the SAIF name A/B/C for the pin InstDecode/U15/AN.

```
prompt> saif_map -set_name A/B/C [get_pins InstDecode/U15/AN]
```

The following example adds the SAIF name A/B/C for the pin InstDecode/U15/AN to the name-mapping database, also indicating that the pin is logically inverted.

```
prompt> saif_map -add_name A/B/C -inverted [get_cells InstDecode/U15/AN]
```

The following example removes the SAIF name A/B for the cell InstDecode/U15.

```
prompt> saif_map -remove_name A/B [get_cells InstDecode/U15]
```

In the following example, the SAIF name mydesign is changed to InstDecode. When a SAIF file containing mydesign is read after this, mydesign is interpreted as InstDecode and annotation is performed accordingly.

```
prompt> saif_map -change_name {mydesign InstDecode}
```

The following example reports the manual name mappings that you specified. Note that the **-set_name** and **-add_name** functions, performed on the pin InstDecode/U15/AN, is applied to its connected net InstDecode/n3, as shown in the report.

```
prompt> saif_map -change_name {abc xyz}
prompt> saif_map -set_name x [get_pin InstDecode/U15/AN]
prompt> saif_map -add_name y [get_pin InstDecode/U15/AN]
prompt> saif_map -report
##########################
# USER SAIF-NAME MAPPING #
##########################
saif_map -change_name {abc xyz}
saif_map -set_name x [get_net InstDecode/n3]
saif_map -set_name y [get_net InstDecode/n3]
```

The following example resets the manual, user-specified SAIF name-mappings.

```
prompt> saif_map -reset
```

The following example returns the SAIF names corresponding to the pin InstDecode/U15/AN.

```
prompt> saif_map -change_name {x c}
prompt> saif_map -set_name a/b/c [get_pins InstDecode/U15/AN] -inverted
prompt> saif_map -add_name c/d/e [get_pins InstDecode/U15/AN]
prompt> saif_map -get_saif_names [get_pins InstDecode/U15/AN]
{{pin InstDecode/U15/AN} {saif_names {c/d/e x/d/e}} {inverted_saif_names a/b/c}}
```

The following example returns the object to which the SAIF name mydesign/bar/blah maps.

```
prompt> saif_map -change_name {mydesign/bar InstDecode/U15}
```

```
prompt> saif_map -set_name InstDecode/U15/blah [get_pin InstDecode/U15/AN]
prompt> get_nets -of_objects [get_pin InstDecode/U15/AN]
InstDecode/n3
prompt> saif_map -get_object_names mydesign/bar/blah
{mydesign/bar/blah {objects {{net InstDecode/n3}}}}
```

SEE ALSO

read_saif(2)

save_block

Saves a block to its current location or as a new block.

SYNTAX

status **save_block**
[-as *block_name*]
[-label *label_name*]
[-force]
[-hierarchical]
[-verbose]
[-compress]
[-blocks *blocks*]
[*block*]

Data Types

block_name string
label_name string
blocks collection
block string

ARGUMENTS

-as *block_name*

Specifies the destination block name with optional library and view specifications. The format for *block_name* is:

[libName:]blockName[/labelName][.viewName]

If no library is specified, **current_lib** is used. If no block name is specified, **current_block** is used. If no view name is specified, the design view is used. If no label name is specified, the default label "" is used. Note that **save_block block_name -as block_name** is equivalent to **save_block block_name**. It is an error to save to a sub-block referenced by the top block. The source and destination label must be different when used with the **-hierarchical** option; otherwise, it is an error.

By default, a new block of the specified block_name is created and saved to disk. This behavior can be changed by setting the "design.morph_on_save_as" application option to true. In this case, the source block and all of its views are renamed into the specified block_name and then saved, instead of making a copy of the source block and saving the copy to disk. The following conditions must be met: 1) the destination and source block are in the same library, and 2) no other blocks in memory reference the source block or any of its views.

-label *label_name*

Specifies the user label to be used when saving the block(s) using this command. This command is the safe way to save a block

(or a physical hierarchy rooted at a top block when combined with the **-hierarchical** option) to a new user label. This is generally used for stages of flow progression, with user label names like "placed", "clocked", and "routed" being typical. The user label names are entirely at the discretion of the user. Specifying an empty user label name causes the block(s) to be saved to the default, which is to have no user label.

As with the **-as** option above, this option is modified by use of the "design.morph_on_save_as" application option, in the same way.

This option is exclusive with the **-as** option.

-force

Overwrites the destination block when the destination is open and modified but not yet saved. This option is used with the **-as** option.

-hierarchical

Saves the entire physical hierarchy of the specified block. When used with the **-as** option and a specific label name, it saves each non library cell reference block and all its views into the specified label. The command follows the hierarchy into the reference blocks and save the subblocks referenced by its instances, stopping when it reaches the leaf-level library cell references. Each labeled block is created in-place in the library of the original block, and each of its non library cell instances are updated such that their references point to the new labeled block.

-verbose

Prints out additional debugging and informational messages.

-compress

Saves design in compressed format. This can be used along with -hierarchical to save the entire physical hierarchy in compressed format. Once a block has been saved with "-compress", the "is_compressed" attribute on the block is set to true. All future save_block on this block will be in compressed format even if the "-compress" option isn't specified. If the "is_compressed" attribute is later set to false, future save_block on this block will be in uncompressed format.

-blocks *blocks*

specifies the source block collection which are to saved. This option is exclusive with options **-as** and **block**. When this option is specified, the blocks corresponding to the collection are saved.

block

Specifies the source block name with optional library and view specifications. The format is as follows:

[libName:]blockName[/labelName][.viewName]

If the library specification is not present, the **current_lib** is used. If the label specification is not present, the default label "" is used. If the view specification is not present, then *design* view is used. If the option is not given, the **current_block** is used.

DESCRIPTION

This command saves the specified (or current) block. When the **-as** option is used, a copy of the block is saved to the specified block destination. The destination block can be in a different library. It can also be with a different block name, or with the same block name by different label name. With option **-blocks**, a collection of blocks can be saved. The command returns 1 if the block is saved and 0 if not. If an illegal name (with a slash) is given for the block, a Tcl error is raised.

Note that the command fails if the block is opened in read-only mode and the **-as** option is not used. The command fails if the **-as** option is used but the destination block exists, but is open for read-only access, or if the destination block has been modified, but not

saved, and the **-force** option is not also used.

When no view is explicitly specified for both source and destination then all available views of source block are saved into destination block.

A given source block view can't be saved into a different destination block view, like a design view can't be saved to a frame view. An error is issued while trying to do such operation.

EXAMPLES

This example saves the block Top and all its views as Top2, meaning that it becomes Top2. Note that Top no longer exists in memory, but remains unchanged on disk.

```
prompt> save_block -as Top2 Top
1
```

This example saves the current block and its particular view.

```
prompt> save_block
1
```

This example saves the block Mid and all its views, which is not the current block:

```
prompt> save_block Mid
1
```

This example does a hierarchical save of Top and all its views, which includes Mid and Bot:

```
prompt> save_block -hierarchical Top
1
```

This example saves the entire physical hierarchy of "Top" into a set of "placed" labeled blocks. This will create a labeled block from "top" and below in their respective libraries, stopping at the library cell level, for each of Top, Mid, and Bot:

```
prompt> save_block Top.design -hierarchical -as Top/placed -verbose
Information: Saving 'Top_lib:Top.design' to 'Top_lib:Top/placed.design'
Information: Saving 'Top_lib:Mid.design' to 'Top_lib:Mid/placed.design'
Information: Saving 'Top_lib:Bot.design' to
"Top_lib:Bot/placed.design"
1
```

This example saves the entire physical hierarchy starting from Mid and all its views into a set of "placed2" label blocks. This will create a labeled block from "Mid" and below, for each of Mid, and Bot:

```
prompt> save_block Mid -hierarchical -as Mid/placed2 -verbose
Information: Saving 'Top_lib:Mid.design' to 'Top_lib:Mid/placed2.design'
Information: Saving 'Top_lib:Bot.design' to
"Top_lib:Bot/placed2.design"
1
```

This example saves just the Top and all its views into a "placed3" labeled block. Note that only the "Top" block is saved as a copy. New label are not created for the lower-level blocks, and "Top/placed" still references "Mid" and "Bot" blocks.

```
prompt> save_block Top -as Top/placed3 -verbose
Information: Saving 'Top_lib:Top.design' to 'Top_lib:Top/placed3.design'
Information: Saving 'Top_lib:Top.frame' to 'Top_lib:Top/placed3.frame'
```

1

SEE ALSO

close_blocks(2)
copy_block(2)
current_block(2)
current_design(2)
get_blocks(2)
get_designs(2)
move_block(2)
open_block(2)
open_lib(2)
remove_blocks(2)
reopen_block(2)

save_drc_error_data

Saves an external DRC error data object to disk.

SYNTAX

status **save_drc_error_data**
drc_error_data
[-as *file_name*]

Data Types

drc_error_data collection
file_name string

ARGUMENTS

drc_error_data

Specifies the collection of external physical DRC error data objects to save.

-as *file_name*

Specifies a new file name for the external error data file.

If you do not specify this option, the error data file is saved with its original name.

DESCRIPTION

The **save_drc_error_data** command saves the specified external DRC error data object to disk.

This command does not save error data objects that are attached to a block. To save an error data object that is attached to a block, save the block with the **save_block** command. To save the attached error data object to a file, use the **write_drc_error_data** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLE

The following example opens the external error data file named my_design_dppinassgn.err, removes fixed errors, and then saves and closes the file.

```
prompt> set data [open_drc_error_data -file_name my_design_dppinassgn.err]
{my_design_dppinassgn.err}
prompt> remove_drc_errors -error_data $data \
  [get_drc_errors -error_data $data -filter {status==fixed}]
prompt> save_drc_error_data $data
1
prompt> close_drc_error_data $data
1
```

SEE ALSO

close_drc_error_data(2)
create_drc_error_data(2)
get_drc_error_data(2)
open_drc_error_data(2)
remove_drc_error_data(2)
save_block(2)
write_drc_error_data(2)

save_ems_database

Saves opened EMS databases to disk.

SYNTAX

status **save_ems_database**
[-as *filename* | -all | *ems_databases*]
[-overwrite]

Data Types

filename string
ems_databases collection

ARGUMENTS

-as *filename*

Saves the current EMS database to disk with the specified name. The *filename* argument must end with ".ems". If there is no current EMS database, then tool will issue an error message. If a file with name "filename" already exists, the command will issue an error message. Use the **-overwrite** option to overwrite an existing file.

-all

Writes all opened EMS databases to disk. If a file already exists on disk with the same name as the EMS database, the command issues an error message. Include the **-overwrite** option to overwrite existing files.

ems_databases

Specifies a collection of EMS databases to write. Use the **get_ems_databases** to create a collection of all EMS databases. If the database files already exist on disk, the command issues an error message. Use the **-overwrite** option to overwrite the files.

-overwrite

Overwrites an existing EMS database file on disk. By default, the command issues an error message if a file with the same name already exists on disk.

DESCRIPTION

The **save_ems_database** command writes the current EMS databases in memory to disk. When no filename is specified, this command writes out the current EMS database. If there is no current EMS database, the command issues a warning message. When you specify a filename, this command writes out the current EMS database to the specified file. The **save_ems_database -all**

command writes out all open EMS databases. **save_ems_database {db1.ems db2.ems}** writes out the EMS databases specified in the collection. The **-all**, **-as** and *ems_databases* options are mutually exclusive.

EXAMPLES

The following examples shows the error message your receive if you run the **save_ems_database** command and there are no open EMS databases.

```
prompt> get_current_ems_database
prompt> save_ems_database
Warning: No current EMS database present. (EMS-036)
```

The following example writes out the current EMS database.

```
save_ems_database with current database
prompt> get_current_ems_database
{test.ems}
prompt> save_ems_database
```

The following example overwrites the test.ems file with the current EMS database.

```
save_ems_database using -as option
prompt> save_ems_database -as test.ems
Error: File 'test.ems' already exists. Use -overwrite flag to force
  writing of messages database. (EMS-002)
prompt> save_ems_database -as test.ems -overwrite
```

The following example writes out all open EMS databases.

```
prompt> get_ems_databases
{test.ems abc.ems}
prompt> save_ems_database -all
```

The following example writes out all open EMS databases; the database names are specified with the **get_ems_databases** command.

```
prompt> get_ems_databases
{test.ems abc.ems}
prompt> save_ems_database [get_ems_databases]
```

SEE ALSO

close_ems_databases(2)
create_ems_database(2)
get_current_ems_database(2)
get_ems_databases(2)
open_ems_database(2)
set_current_ems_database(2)

save_lib

Saves a library and its changed blocks to disk.

SYNTAX

status **save_lib**
[-as *library_name* [-version *release_version*]]
[-all]
[-compress]
[*library*]

Data Types

library_name string
release_version string
library collection

ARGUMENTS

-as *library_name*

Specifies the name of the saved library.

If you do not specify this option, the library is saved with the same name.

This option is mutually exclusive with the **-all** option; you can specify only one of these options.

-version *release_version*

Specifies the version of the saved library.

To see the valid version strings, use the **report_versions** command.

This option is valid only with the **-as** option.

-all

Saves all open libraries that have changed blocks. Note that in an implementation tool, the command saves only design libraries and not reference libraries that contain library cells.

This option is mutually exclusive with the **-as** option and *library* argument; you can specify only one of these arguments.

-compress

Saves libraries in compressed format. This option is applicable to design libraries only. lib-cell libraries cannot be compressed. Once a library has been saved with "-compress", the "is_compressed" attribute on the library is set to true. All future save_lib or save_block on this library or its blocks will be in compressed format, even if the "-compress" option isn't specified. If the

"is_compressed" attribute is set to false, future save_lib on this library will be in uncompressed format.

library

Specifies the library to save. This can be either the library name or a collection that contains a single library.

If you do not specify this option, the command saves the current library.

This argument is mutually exclusive with the **-all** option; you can specify only one of these arguments.

DESCRIPTION

This command saves the library data to disk. Library data includes the design catalog, library attributes, technology data, and parasitic data. The command also saves all blocks in the library that have been modified but not saved.

In an implementation tool, the command saves only design libraries in edit mode. It does not save read-only libraries or reference libraries that contain library cells.

If the command successfully saves one or more libraries, it returns 1. Otherwise, it returns 0. If you try to save a library with an illegal name, such as a name that contains a slash, the command raises a Tcl error.

EXAMPLES

This example saves the contents of the MyDesign design library.

```
prompt> save_lib MyDesign
1
```

This example saves the contents of the Top library to a library named Top.1.0 using the database schema revision for version J-2014.06.

```
prompt> save_lib Top -as Top.1.0 -version J-2014.06
Saving library 'Top' as 'Top.1.0' in version 'J-2014.06'
1
```

SEE ALSO

create_lib(2)
open_lib(2)
close_lib(2)
copy_lib(2)
move_lib(2)
current_lib(2)
get_libs(2)
set_ref_libs(2)
search_path(3)
report_versions(2)

save_upf

Writes out the UPF commands in the specified file.

SYNTAX

collection **save_upf**
[-full_chip | -for_empty_blackbox]
[-format supplemental]
[-force_reference | -force_no_reference]
[-include | -exclude]
[-nosplit]
file_name

Data Types

file_name string

ARGUMENTS

-format supplemental

Specifies the format of output file. Specifying *supplemental* means that the output file contains only the UPF constraints generated by the tool(s). This option and value work only in the golden UPF mode.

For the golden UPF flow, when the application option **mv.upf.enable_golden_upf** is set to true, default is set to *supplemental* since only the supplemental format is supported. So specifying **-format** is not really necessary.

-full_chip

Save full_chip UPF from top-only design links with blocks. The linked blocks can be abstract view (recommended) or design view.

-for_empty_blackbox

Save UPF for empty black_box created from this design. This option cannot be combined with **-format**.

-force_reference

Specifies lib_cells to output for **connect_supply_net** or **set_related_supply_net** if exists. This option cannot be combined with **-full_chip**, **-for_empty_blackbox**.

-force_no_reference

Specifies lib_cells to not output for **connect_supply_net** or **set_related_supply_net** if exists. This option cannot be combined with **-full_chip**, **-for_empty_blackbox**.

-include

Types of cell instance to include in **connect_supply_net** or **set_related_supply_net**. This option cannot be combined with **-exclude**, **-full_chip**, **-for_empty_blackbox**.

-exclude

Types of cell instance to exclude in **connect_supply_net** or **set_related_supply_net**. This option cannot be combined with **-include**, **-full_chip**, **-for_empty_blackbox**.

-nosplit

Specifies the UPF file to be output in none split mode, the commands will be output in a line instead of multiple lines. By default, user UPF will preserve the line width as specified in input UPF file, derived UPF will be output as the values defined by app option mv.upf.save_upf_line_width and mv.upf.save_upf_line_indent.

It is exclusive with option *-full_chip*.

file_name

Specifies the name of the file to which UPF commands are to be written out.

DESCRIPTION

This command writes out the UPF file that is currently part of the design to the specified file. This UPF file is what you have loaded by using the **load_upf** command or the command line, with the changes made to the file during optimization, and the UPF commands specified at the command prompt in that session.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following is an example of the **save_upf** command:

```
prompt> save_upf design.upf
```

The user section in the UPF file will preserve the line width as specified in input UPF file, the derived section in the file will be output with the default value of line width, and the continuing lines for a command in multiple lines will be indented with the default value of line indent.

Following are an example for command option *-nosplit* and app options

mv.upf.save_upf_line_width and *mv.upf.save_upf_line_indent*, they will impact the UPF file as below.

prompt> **save_upf -nosplit design.upf**

The commands both in user and derived section will be printed in a line instead of multiple lines.

Following are another example for line width and line indent.

```
prompt> set_app_options -name mv.upf.save_upf_line_width -value 60
prompt> set_app_options -name mv.upf.save_upf_line_indent -value 2
prompt> save_upf design.upf
```

The commands both in user and derived section will be printed in multiple lines if the command reaches max line width 60, and continuing command lines will be indented with 2 empty spaces.

SEE ALSO

load_upf(2)
mv.upf.enable_golden_upf(3)
mv.upf.save_upf_line_width(3)
mv.upf.save_upf_line_indent(3)

send_status

Sends the specified message from the distributed processing machine to the calling process.

SYNTAX

status **send_status**
-status *message_string*

Data Types

message_string string

ARGUMENTS

-status *msg_string*

Specifies the string to send to the main process.

DESCRIPTION

This command is used only by a distributed processing worker to send a text string to the main process. Distributed processing is initiated by using the **run_block_script** command. You can add this command to the distributed processing script to send status, runtime, or any other messages to the main calling process.

EXAMPLES

The following example sends a message from the distributed processing worker to the main process. The test.tcl distributed processing script contains the following command:

```
send_status -status "Processing block $block_refname"
```

The script is called by the **run_block_script** command as follows and produces the output shown.

```
prompt> set_host_options -name myLSFtest -submit_command "bsub"
1
prompt> run_block_script -script test.tcl -host_options myLSFtest -ref_blocks { A B }
...
Processing block A
```

```
Processing block B
1
```

SEE ALSO

run_block_script(2)
set_host_options(2)

set_analyze_rtl_logic_level_threshold

Specifies the logic level threshold value to be used by report_logic_levels command. Also used to remove the previously set value.

SYNTAX

status **set_analyze_rtl_logic_level_threshold**
[-group <group_name>]
[-reset]
[threshold]

Data Types

group_name string
threshold integer

ARGUMENTS

-group *group_name*

Specifies a name for the path group for which threshold value is being set or reset. The group name should be one of the group names reported by report_path_group command. When this option is not used, the threshold setting is applied at global level.

-reset

Specifies that the previously specified threshold value be reset. If -group option is specified, then reset applies only to the specified path group. If -group option is not specified, then the threshold value is reset for all path groups and global level.

threshold

Specifies the threshold integer value to be used by report_logic_levels command.

DESCRIPTION

This command is used to specifiy the threshold value to be used by report_logic_levels command. The value can be specified at each path group level or at global level. When threshold is specified both at global level and path group level, path group threshold setting is used for the paths belonging to that path group.

Not using this command will result in tool computing the threshold automatically based on the required time for the path and the delay of average size NAND2 cell in the target library.

When threshold is specified both at global level and path group level and reset is applied on path group level, global threshold setting is used for paths belonging to that path group.

When threshold is specified only at path group level and reset is applied on that path group level, in absence of global threshold, tool computed threshold is used for paths belonging to that path group.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the logic level threshold to 10 at global level.

```
prompt> set_analyze_rtl_logic_level_threshold  10
```

The following example sets the logic level threshold to 5 only for the paths in the path group CLOCK while setting 6 for all other paths.

```
prompt> set_analyze_rtl_logic_level_threshold 6
prompt> set_analyze_rtl_logic_level_threshold -group CLOCK 5
```

The following example re-sets the logic level threshold for path group CLOCK.

```
prompt> set_analyze_rtl_logic_level_threshold -group CLOCK -reset
```

The following example re-sets the logic level threshold for all paths (all path groups).

```
prompt> set_analyze_rtl_logic_level_threshold -reset
```

SEE ALSO

report_logic_levels(2)
report_path_group(2)

set_annotated_check

Sets the setup, hold, recovery, or removal timing check value between two pins.

SYNTAX

status **set_annotated_check**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
check_value
-from *from_pins*
-to *to_pins*
-setup | -hold | -recovery | -removal | -nochange_high | -nochange_low | -width | -period
[-rise | -fall]
[-clock rise | fall]
[-increment]
[-override_increment]

Data Types

check_value float
from_pins list
to_pins list

ARGUMENTS

-modes *mode_list*

Specifies the modes for which to apply the annotated value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is annotated separately. If this option is not given, the command applies annotation for every scenario that is active with the specified corner(s). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to apply the annotated value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is annotated separately. If this option is not given, the command applies annotation for every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios for which to apply the annotated value. Each of the specified scenarios is annotated separately. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command applies to the current scenario only.

check_value

Specifies the timing check value between pins on the same cell, which can be positive or negative. You must express the *check_value* argument in units consistent with the technology library used during optimization. For example, if the technology library specifies delay values in nanoseconds, the *check_value* argument must be expressed in nanoseconds.

-from *from_pins*

Specifies the leaf-cell clock pins that are the startpoints of the timing arcs for which checks are to be annotated.

-to *to_pins*

Specifies the leaf-cell data pins that are the endpoints of the timing arcs for which checks are to be annotated.

-setup | **-hold** | **-recovery** | **-removal** | **-nochange_high** | **-nochange_low** | **-width** | **-period**

Specifies the type of the timing check. There must be a corresponding timing arc between the from and to pins.

- The **-setup** option specifies that data must be stable for the amount of time specified by *check_value* before the closing clock edge.
- The **-hold** option specifies that data must be stable for the amount of time specified by *check_value* after the closing clock edge.
- The **-recovery** option specifies that an asynchronous set or clear inactive edge cannot occur within the amount of time specified by *check_value* before the closing clock edge. This is essentially a setup requirement on an asynchronous pin, but only the inactive transition is considered.
- The **-removal** option specifies that an asynchronous set or clear inactive edge cannot occur until the amount of time specified by *check_value* after the closing clock edge. This is essentially a hold requirement on an asynchronous pin, but only the inactive transition is considered.
- The **-nochange_high** option provides a timing check that specifies that the data must not rise within the amount of time specified by *check_value* before the clock becomes active and must not fall until the amount of time specified by *check_value* after the clock becomes inactive.
- The **-nochange_low** option provides a timing check that specifies that the data must not fall within the amount of time specified by *check_value* before the clock becomes active and must not rise until the amount of time specified by *check_value* after the clock becomes inactive.
- The **-width** option specifies a minimum pulse width constraint for a clock pin. The timing check is defined on the -from pin, so the related pin (-to) should be the same.
- The **-period** option specifies the minimum period constraint for a clock pin. The timing check is defined on the **-from** pin, so the related pin (**-to**) should either be the same or it can be omitted.

-rise | **-fall**

Specifies whether the check is for the data rise or fall transition. If you do not specify either **-rise** or **-fall**, both values are set.

The **-rise** option specifies a rise data transition that corresponds to the check before the activating clock edge. A rise data transition corresponds to the check after the deactivating clock edge. A rise clock transition indicates that the clock is active-high.

The **-fall** option specifies a fall data transition that corresponds to the check after the deactivating clock edge. A fall data transition corresponds to the check before the activating clock edge. A fall clock transition indicates that the clock is active-low.

-clock rise | **fall**

Specifies whether the check is for clock rising or falling. By default, checks for both clock rise and fall are set.

-increment

Adds the specified check value to the existing default check value. Without this option, the check value replaces the default check value. If you use the **set_annotated_check** command multiple times with the **-increment** option, the last **-increment** setting is

used and earlier ones are ignored. If you use a **set_annotated_check** command without the **-increment** option, that check value is used, and any **-increment** setting is ignored.

-override_increment

Overwrite incremental values instead of adding them to the already existing ones. Applies only for annotating incremental values.

DESCRIPTION

This command annotates a timing check between two or more pins on a cell or a net in the current design. This might be appropriate after place and route for technologies where the timing check value varies for different instances and the library timing check values do not provide sufficient accuracy.

If the design is not already linked, the **set_annotated_check** command links it automatically.

The command can be used for pins at lower levels of the design hierarchy. Pins are specified as "INSTANCE1/INSTANCE2/PIN_NAME."

To list annotated timing check values, use the **report_annotated_check** command. To remove annotated timing check values from a design, use the **remove_annotated_check** or **reset_design** command. To see the effect of **set_annotated_check** for a specific instance, use the **report_timing** command.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example annotates a setup time of 2.1 units between the CP clock pin and the D data pin of the u1/ff12 cell instance:

```
prompt> set_annotated_check -setup 2.1 \
  -from u1/ff12/CP -to u1/ff12/D
```

The following example annotates a nochange check between data low and clock low. The nochange margin before the clock is 1.3 units and the nochange margin after the clock is 2.4 units.

```
prompt> set_annotated_check -nochange_low 1.3 \
  -clock fall -fall -from u1/CP -to u1/EN

prompt> set_annotated_check -nochange_low 2.4 \
  -clock fall -rise -from u1/CP -to u1/EN
```

SEE ALSO

current_design(2)
link(2)
remove_annotated_check(2)

report_annotated_check(2)
report_timing(2)
reset_design(2)

set_annotated_delay

Sets the net or cell delay value between two pins.

SYNTAX

status **set_annotated_delay**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
-net | -cell
[-rise | -fall]
[-min]
[-max]
[-increment]
delay_value
-from *from_pins*
-to *to_pins*

Data Types

delay_value	float
from_pins	list
to_pins	list

ARGUMENTS

-modes *mode_list*

Specifies the modes for which to apply the annotated value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is annotated separately. If this option is not given, the command applies annotation for every scenario that is active with the specified corner(s). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to apply the annotated value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is annotated separately. If this option is not given, the command applies annotation for every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios for which to apply the annotated value. Each of the specified scenarios is annotated separately. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command applies to the current scenario only.

-net | -cell

Specifies whether the delay annotated is a net or cell delay. You must specify one of the options, but not both.

-rise | **-fall**

Specifies whether the delay is for data rise or fall transition. If you do not specify **-rise** or **-fall**, both values are set.

-min

Specifies that the delay is to be used for minimum delay analysis.

-max

Specifies that the delay is to be used for maximum delay analysis. By default, the delay is used for maximum and minimum delay analysis.

-increment

Increments the delay value to the current delay of the specified timing arc.

delay_value

Specifies the delay value between pins on the same cell or net. The *delay_value* must be expressed in units consistent with the logic library used during optimization. For example, if the logic library specifies delay values in nanoseconds, *delay_value* must be expressed in nanoseconds.

-from ***from_pins***

Specifies a list of leaf-cell pins or top-level ports that are the startpoints of the timing arcs for which delays are to be annotated.

-to ***to_pins***

Specifies a list of leaf-cell pins or top-level ports that are the endpoints of the timing arcs for which delays are to be annotated.

DESCRIPTION

This command annotates cell or net delays in the design.

Cell delays exist between pins of the same leaf cell. With the **-cell** option, pins in *from_pins* must be cell input or inout pins, and pins in *to_pins* must be cell output or inout pins.

Net delays exist between leaf-cell pins or top-level ports connected by a net. With the **-net** option, pins in *from_pins* must be cell output or inout pins or top-level input ports, and pins in *to_pins* must be cell input or inout pins or top-level output ports.

To verify the accuracy of the back-annotation done by the **set_annotated_delay** command, run the **update_timing** command.

The specified delay value overrides the internally-estimated cell and net delay value. If the specified pins are not in the same cell or on the same net, when the timing is updated, the tool issues an error message and *delay_value* is discarded for those pins.

You can use **set_annotated_delay** for pins at lower levels of the design hierarchy. Pins are specified as "INSTANCE1/INSTANCE2/PIN_NAME."

To remove the annotated cell or net delay values from a design, use the **remove_annotated_delay** or **reset_design** command.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example annotates a cell delay of 20 units between input pin A of cell instance U1/U2/U3 and output pin Z of the same cell instance.

```
prompt> set_annotated_delay -cell 20 \
    -from U1/U2/U3/A -to U1/U2/U3/Z
```

The following example annotates a rise net delay of 1.4 units between output pin *U1/Z* and input pin *U2/A*.

```
prompt> set_annotated_delay -net -rise 1.4 \
    -from U1/Z -to U2/A
```

SEE ALSO

remove_annotated_delay(2)
report_timing(2)
reset_design(2)

set_annotated_power

Sets cell power dissipation values. Overrides calculated values.

SYNTAX

status **set_annotated_power**
-scenarios *scenario_list*
-supply_net *supply_net*
-leakage *power_value*
-internal *power_value*
cell_list

Data Types

power_value float
scenario_list list
cell_list list

ARGUMENTS

-scenarios

Specifies the set of scenarios for which the annotated values apply. The default is the current_scenario value.

-supply_net

Specifies the supply net from which the specified power is drawn. You can annotate power for multiple supply nets by multiple invocations of the command.

-leakage

Specifies the leakage power value in user power units. The -leakage option can be used together with the -internal option.

-internal

Specifies the internal (dynamic) power value in user power units. The -internal option can used together with the -leakage option.

cell_list

Specifies the cells for which the power values apply. The command can be used to annotate black-box cells, physical block instances or leaf cells.

DESCRIPTION

This command annotates leakage and/or internal power values on cells. The annotated power overrides calculated power values. If a leakage value is annotated, **all** calculated leakage values for the specified scenarios are discarded. Likewise, if an internal power value is annotated **all** calculated internal power values for the scenarios are discarded.

The cells to be annotated should be either black-box cells, physical block instances or leaf cells.

If no scenario is specified, the values are applied for the current scenario only. If no supply net is specified, for leaf cells the primary supply net for the domain of the cell is used. For hierarchical cells, no supply net will be infered when no *-supply_net* option was specified.

The power is accounted in the power_group of the cell. If the cell has no power group (e.g. for physical blocks), implicit power_group **annotated** is used.

Multicorner-Multimode Support

EXAMPLES

The following example annotates a cell leakage power of 13.2 user units on cell instance U1/U2/U3 for the current scenario and the primary supply net of the cell's domain. The annotated leakage power is accounted in the power_group of the cell. All calculated leakage power of the cell in the current scenario is discarded.

```
prompt> set_annotated_power -leakage 13.2 \
        U1/U2/U3
```

The following example annotates a cell leakage power of 13.2 user units and an internal power of 125.6 user units on cell instance U1/U2/U3 for scenario *func.wc* and supply net *VDDB*. All calculated leakage power and internal power of the cell in scenario *func.wc*. is discarded. The annotated leakage power is accounted in the power_group of the cell.

```
prompt> set_annotated_power -leakage 13.2 \
        -internal 125.6 -scenarios func.wc \
        -supply_net VDDB U1/U2/U3
```

The following example annotates a cell leakage power of 13.2 user units and an internal power of 125.6 user units on cell instance U1/U2/U3 for scenario *func.wc* and supply net *VDD*. For supply net *VDDB* it annotates an internal power of 30.4 in scenario *func.wc*. All calculated leakage power and internal power of the cell in scenario *func.wc*. is discarded. The annotated leakage power is accounted in the power_group of the cell.

```
prompt> set_annotated_power -leakage 13.2 \
        -internal 125.6 -scenarios func.wc \
        -supply_net VDD U1/U2/U3
prompt> set_annotated_power \
        -internal 30.4 -scenarios func.wc \
        -supply_net VDDB U1/U2/U3
```

SEE ALSO

remove_annotated_power(2)
report_annotated_power(2)

set_annotated_transition

Sets the transition time at a given pin.

SYNTAX

status **set_annotated_transition**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-rise | -fall]
[-min]
[-max]
transition
port_pin_list

Data Types

transition	float
port_pin_list	list

ARGUMENTS

-modes *mode_list*

Specifies the modes for which to apply the annotated value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is annotated separately. If this option is not given, the command applies annotation for every scenario that is active with the specified corner(s). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to apply the annotated value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is annotated separately. If this option is not given, the command applies annotation for every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios for which to apply the annotated value. Each of the specified scenarios is annotated separately. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command applies to the current scenario only.

-rise | **-fall**

Specifies whether the transition time is for a data rise or data fall transition. If you do not specify **-rise** or **-fall**, both values are set.

-min

Specifies that the transition is to be used for minimum delay analysis. By default, the transition value is used for maximum and minimum delay analysis. If a transition value is annotated for only minimum or maximum delay analysis, that value is used for both maximum and minimum delays. It is possible to annotate different transition values for minimum and maximum analysis, but it is not possible to annotate only for minimum or annotate only for maximum.

-max

Specifies that the transition is to be used for maximum delay analysis. By default, the transition value is used for maximum and minimum delay analysis. If a transition value is annotated for only minimum or maximum delay analysis, that value is used for both maximum and minimum delays. It is possible to annotate different transition values for minimum and maximum analysis, but it is not possible to annotate only for minimum or annotate only for maximum.

transition

Specifies the transition value at the pins supplied with the *port_pin_list* argument. The transition value must be expressed in units consistent with the technology library used during optimization. For example, if the technology library specifies transition values in nanoseconds, the transition values must be expressed in nanoseconds.

port_pin_list

Specifies a list of leaf-cell pins or top-level ports that are the endpoints of the timing arcs for which delays are to be annotated.

DESCRIPTION

This command sets the transition time at a given pin.

To remove the annotated transition values from a design, use the **remove_annotated_transition** or **reset_design** command.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example annotates a transition time of 20 units at input pin *A* of cell instance U1/U2/U3:

```
prompt> set_annotated_transition 20  U1/U2/U3/A
```

The following example annotates a rise transition of 1.4 units at the input pin U5/A:

```
prompt> set_annotated_transition -rise 1.4 U5/A
```

The following example annotates a rise transition of 12.3 units for minimum delay analysis at U5/A:

```
prompt> set_annotated_transition -rise -min 12.3 U5/A
```

SEE ALSO

current_design(2)
remove_annotated_transition(2)
report_timing(2)
reset_design(2)

set_aocvm_coefficient

Sets advanced on-chip variation (OCV) random coefficients on cells and library cells for use during an advanced OCV analysis.

SYNTAX

status **set_aocvm_coefficient**
rnd_coeff
object_list

Data Types

rnd_coeff float
object_list list

ARGUMENTS

rnd_coeff

Specifies the random coefficient applied to the objects in the *object_list*. The *rnd_coeff* setting must be a floating-point value greater than zero.

object_list

Specifies a list of library cells or leaf cells on which the coefficient is set.

DESCRIPTION

This command sets random coefficients on library cells or leaf cells for advanced on-chip variation (OCV) analysis. These are user-specified depth adjustment factors (default of 1.0) that you can apply at the library cell or cell instance level. These coefficients are not required for an advanced OCV analysis. However, their application may increase the accuracy of the analysis.

If different coefficients have been annotated on a library cell and an instance of that cell, the coefficient annotated on the cell instance takes precedence.

Random coefficients are used to calculate cell path depths. The default increment of path depth for any cell is 1.0. The path depth increment for a cell, if defined, is *rnd_coeff*.

To display the advanced OCV coefficients that have been set, use the **report_ocvm** command.

To remove advanced OCV coefficients from a cell, use the **remove_ocvm -coefficient** command.

Multicorner-Multimode Support

EXAMPLES

The following example sets an advanced OCV random coefficient on a library cell named *lib/bufA*:

```
prompt> set_aocvm_coefficient 1.2 [get_lib_cells lib/bufA]
```

SEE ALSO

read_ocvm(2)
remove_ocvm(2)
report_ocvm(2)

set_app_options

Sets application options to the specified value on the specified block.

SYNTAX

list **set_app_options**
[-block *block*]
[-as_user_default]
[-category *category*]
[-name *name*]
[-value *value*]
[-list *name_value_list*]

Data Types

block	block name or collection
category	string
name	string
value	string
name_value_list	list

ARGUMENTS

-block *block*

The name of the block on which to set the option values.

If this option is not specified, the option value is set on the current block, if the option is a block-scoped application option.

This option cannot be specified with the **-as_user_default** option.

-as_user_default

Sets the specified value as the user default for the application option.

Note that the user default setting applies only to the current session and is not saved.

This option cannot be specified with the **-block** option.

-category *category*

Specifies one or two levels of the category of the application options to follow, either with the **-name** or **-list** option. When setting multiple application options that belong to the same category, this option enables you to provide the category one time, instead of multiple times with the option name.

If this option is not specified, you must specify the complete application option name.

-name ***name***

Specifies the name of the application option to modify.

The **-name** and **-value** options go together and are mutually exclusive with the **-list** option. You must specify either the **-name** and **-value** options or specify the **-list** option.

-value ***value***

Specifies the value of the application option specified in the **-name** option.

The **-name** and **-value** options go together and are mutually exclusive with the **-list** option. You must specify either the **-name** and **-value** options or specify the **-list** option.

-list ***name_value_list***

Specifies a list of option name and value pairs to set. The list must have an even number of tokens to be valid. If any of the key value pairs is invalid, this command fails with no effect.

The **-name** and **-value** options go together and are mutually exclusive with the **-list** option. You must specify either the **-name** and **-value** options or specify the **-list** option.

DESCRIPTION

This command sets the values of the specified application options.

There are two types of application options:

- Global application options, which apply everywhere within the current session

  Global application option settings are not stored and are not carried over from one session to the next. To retain these option settings, you must set them in your setup file.

- Block-level application options, which apply only to the blocks on which they are set

  Block-level application option settings are saved with the block in the design library and are persistent across tool sessions except when they are set by the **set_app_options -as_user_default** command. In this case, the application options are treated like global application options and are not persistent across tool sessions.

To determine whether an application option is global or block-scoped, use the **report_app_options** command.

For application options that are of type "list of {name value} pairs", the command supports pointed modifications through the subscript mechanism. For subscripts to be recognized, the application options must be enabled for subscript modification and access.

Many application options have a value that must be expressed in units of time, voltage, and so on. In this case, the value must include a multiplier character (such as 'p', 'n', or 'k') and/or a units specifier character (such as 'F', 'V', or 's'). The units specifier character must use the correct case. For example, uppercase 'V' for volts and lowercase 's' for seconds. If the units character does not specify the correct type of unit, the command fails with no effect. Note that the user input units currently in effect are **not** used by this command.

The command returns the name-value pair of the application option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the options.

EXAMPLES

The following example sets the value of the **time.enable_preset_clear_arcs** application option to **true** on the current block.

```
prompt> set_app_options -name time.enable_preset_clear_arcs -value true
time.enable_preset_clear_arcs true
```

The following example sets the value of the **shell.tmp_dir_path** application option to "/tmp" and the **time.enable_preset_clear_arcs** application option to **true**.

```
prompt> set_app_options \
  -list {shell.tmp_dir_path "/tmp" time.enable_preset_clear_arcs true}
shell.tmp_dir_path /tmp time.enable_preset_clear_arcs true
```

The following example sets the **time.high_fanout_net_pin_capacitance** application option, which requires a capacitance value, to 1.5 picofarads.

```
prompt> set_app_options \
  -name time.high_fanout_net_pin_capacitance -value 1.5p
time.high_fanout_net_pin_capacitance 1.5pF
```

The following example also sets the **time.high_fanout_net_pin_capacitance** application option to 1.5 picofarads.

```
prompt> set_app_options \
  -name time.high_fanout_net_pin_capacitance -value 1.5pF
time.high_fanout_net_pin_capacitance 1.5pF
```

The following example issues an error message because no units or multiplier characters are specified.

```
prompt> set_app_options \
  -name time.high_fanout_net_pin_capacitance -value 1.5
Error: Application option 'time.high_fanout_net_pin_capacitance' must
be specified with a capacitance unit. (UIM-009)
Error: Problem in set_app_options
  Use error_info for more info. (CMD-013
```

The following example issues an error message because the units character is incorrect. Capacitance is represented by an uppercase 'F'.

```
prompt> set_app_options \
  -name time.high_fanout_net_pin_capacitance -value 1.5pf
Error: Invalid value '1.5pf' specified for option
 'time.high_fanout_net_pin_capacitance'
      Use error_info for more info. (CMD-013)
```

The following example issues an error message because the wrong units as specified.

```
prompt> set_app_options \
  -name time.high_fanout_net_pin_capacitance -value 1.5mV
Error: Invalid value '1.5mV' specified for option
 'time.high_fanout_net_pin_capacitance'
       Use error_info for more info. (CMD-013)
```

The following example adds a value for the M1 key of the **test.layer_name_number_pair_list** application option.

```
prompt> set_app_options \
  -name test.layer_name_number_pair_list(M1) -value 31
```

```
test.layer_name_number_pair_list {M1 31}
```

The following example adds a value for the M2 key of the **test.layer_name_number_pair_list** application option.

```
prompt> set_app_options \
  -name test.layer_name_number_pair_list(M2) -value 32
test.layer_name_number_pair_list {M1 31 M2 32}
```

The following example modifies the value of the M1 key of the **test.layer_name_number_pair_list** application option.

```
prompt> set_app_options \
  -name test.layer_name_number_pair_list(M1) -value 33
test.layer_name_number_pair_list {M1 33 M2 32}
```

The following example uses the **-category** option to modify multiple application options of the same category.

```
prompt> set_app_options -category signoff.create_metal_fill \
  -list {flat true run_dir /tmp}
signoff.create_metal_fill.flat true signoff.create_metal_fill.run_dir /tmp
```

SEE ALSO

get_app_option_value(2)
get_app_options(2)
help_app_options(2)
report_app_options(2)
reset_app_options(2)

set_app_var

Sets the value of an application variable.

SYNTAX

string **set_app_var**
-default
var
value

Data Types

var string
value string

ARGUMENTS

-default

Resets the variable to its default value.

var

Specifies the application variable to set.

value

Specifies the value to which the variable is to be set.

DESCRIPTION

The **set_app_var** command sets the specified application variable. This command sets the variable to its default value or to a new value you specify.

This command returns the new value of the variable if setting the variable was successful. If the application variable could not be set, then an error is returned indicating the reason for the failure.

Reasons for failure include:

- The specified variable name is not an application variable, unless the application variable **sh_allow_tcl_with_set_app_var** is set to true See the **sh_allow_tcl_with_set_app_var** man page for details.

- The specified application variable is read only.
- The value specified is not a legal value for this application variable

EXAMPLES

The following example attempts to set a read-only application variable:

```
prompt> set_app_var synopsys_root /tmp
Error: can't set "synopsys_root": variable is read-only
    Use error_info for more info. (CMD-013)
```

In this example, the application variable name is entered incorrectly, which generates an error message:

```
prompt> set_app_var sh_enabel_page_mode 1
Error: "sh_enabel_page_mode" is not an application variable
    Use error_info for more info. (CMD-013)
```

This example shows the variable name entered correctly:

```
prompt> set_app_var sh_enable_page_mode 1
1
```

This example resets the variable to its default value:

```
prompt> set_app_var sh_enable_page_mode -default
0
```

SEE ALSO

get_app_var(2)
report_app_var(2)
write_app_var(2)

set_attribute

Sets attribute values to the specified value on the specified list of objects.

SYNTAX

collection **set_attribute**
[-class *class_name*]
[-quiet]
-objects *object_list*
-name *attribute_name*
-value *attribute_value*
pos_object_list
pos_attribute_name
pos_attribute_value

class_name string
object_list list
attribute_name string
attribute_value string
pos_object_list list
pos_attribute_name string
pos_attribute_value string

ARGUMENTS

-class *class_name*

Specifies the first class object type name used for implicit search of objects. This can be used when objects are specified as string and its object type is not among the predefined types in the search order set by app option *shell.common.implicit_find_mode*.

-quiet

Turns off the warning message that would otherwise be issued if the attribute or objects are not found.

-objects *object_list*

Specifies a list of objects on which the attribute value is to be set. Each element in the list is a collection of objects. This option is mutually exclusive with *pos_object_list*. Either one (and only one) of *-objects* or *pos_object_list* must be specified.

-name *attribute_name*

Specifies the name of the attribute to be set. This option is mutually exclusive with *pos_attribute_name*. Either one (and only one) of *-name* or *pos_attribute_name* must be specified.

-value *attribute_value*

Specifies the value of the attribute. This option is mutually exclusive with *pos_attribute_value*. Either one (and only one) of *-value* or *pos_attribute_value* must be specified.

pos_object_list

Specifies a list of objects on which the attribute value is to be set. Each element in the list is a collection of objects. This positional option is provided for compatibility with other tools.

pos_attribute_name

Specifies the name of the attribute to be set. This positional option is provided for compatibility with other tools.

pos_attribute_value

Specifies the value of the attribute. This positional option is provided for compatibility with other tools.

DESCRIPTION

This command sets the attribute values on the specified objects. This command returns a collection of objects that have the specified attribute value set.

The *object_list* can be specified as strings in which case objects will be implicitly searched based on object type search order as specified by app option *shell.common.implicit_find_mode*. If none of the predefined object type matches specified input string then the command will fail. If the object type is known then user can use *-class* to provide that input.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the attributes. However, for attributes such as **max_leakage_power**, which are scenario dependent, this command uses information from the current scenario only.

EXAMPLES

The following example defines an integer cell attribute named X, and then sets this attribute to 30 on all cells in this level of the hierarchy:

```
prompt> define_user_attribute -type int -classes cell -name X
1
prompt> set_attribute -objects [get_cells *] -name X -value 30
{"U1"}
```

The following example sets the value of *is_fixed* attribute for voltage area VA1 by specifying VA1 as string and using *-class* option since voltage area is not among predefined object types for implicit find search order.

```
prompt> set_attribute -objects VA1 -name is_fixed -value true -class voltage_area
{VA1}
```

The following example sets the value of *orientation* attribute for cell U1234 by specifying U1234 as string.

```
prompt> set_attribute -objects U1234 -name orientation -value R180
{U1234}
```

SEE ALSO

define_user_attribute(2)
get_attribute(2)
get_defined_attributes(2)
list_attributes(2)
remove_attributes(2)
report_attributes(2)
shell.common.implicit_find_mode(3)

set_auto_disable_drc_nets

Disables DRC for specified nets.

SYNTAX

status **set_auto_disable_drc_nets**
[-none]
[-all]
[-constant true | false]
[-scan true | false]
[-on_clock_network true | false]
[-dft_signal signal_list]

ARGUMENTS

-none

Enables DRC for all nets in the current design, including nets connected to clocks.

-all

Disables DRC on all applicable nets in the current design. It is equivalent to setting **-on_clock_network true**, **-constant true**, and **-scan true**.

-constant true | **false**

Disables or enables DRC on constant nets in the current design. When set to **true**, it automatically disables DRC on constant nets. When set to **false**, it enables DRC on constant nets.

-scan true | **false**

Disables or enables DRC on scan enable and scan clock nets in the current design. When set to **true**, it automatically disables DRC on scan enable and scan clock nets. When **false**, it enables DRC on scan enable and scan clock nets.

-scan true | **false**

Disables or enables DRC on scan enable and scan clock nets in the current design. When set to **true**, it automatically disables DRC on scan enable and scan clock nets. When **false**, it enables DRC on scan enable and scan clock nets.

-dft_signal { type1 type2 ... }

Disables DRC only for the specified scan signal types when **-scan true**. If not specified then DRC is disabled for all supported types when **-scan true** is set. The supported types are:

- **ScanEnable**
- **TestMode**

- **wrp_shift**
- **input_wrp_shift**
- **output_wrp_shift**
- **LOSPipelineEnable**
- **ScanClock**

-on_clock_network true | **false**

Disables or enables DRC on clock networks in the current design. When set to **true**, DRC is disabled on clock networks.

DESCRIPTION

This command disables design rule checking (DRC) for specified nets in the current design. The command options are additive each time a new **set_auto_disable_drc_nets** command is executed.

To disable DRC for clock networks, use the **-on_clock_network true** option.

To prevent constant nets from having DRC disabled, use the **-constant false** or **-none** options.

DRC disabled nets are those nets that are free from the max_capacitance, max_fanout, and max_transition design rule constraints. They are useful for reducing DRC violations caused by clock trees, because these nets usually have high max_capacitance and max_fanout violations.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example disables DRC on nets connected to clocks and constant nets in the current design:

prompt> **set_auto_disable_drc_nets -on_clock_network true -constant true**

The following example disables DRC on only the constant nets in the current design:

prompt> **set_auto_disable_drc_nets -constant true**

The following example re-enables DRC for all relevant nets, including those on clock nets, in the current design:

prompt> **set_auto_disable_drc_nets -none**

The following example disables DRC only for ScanEnable and TestMode scan signal nets:

prompt> **set_auto_disable_drc_nets -scan true -dft_signal { ScanEnable TestMode }**

set_auto_floorplan_constraints

Sets constraints for implicit floorplan initialization.

SYNTAX

status **set_auto_floorplan_constraints**
[-control_type core | die]
[-core_utilization *utilization*]
[-shape R | L | T | U]
[-orientation N | W | S | E]
[-side_ratio {side_a side_b [side_c side_d side_e side_f]}]
[-side_length {side_a side_b [side_c side_d side_e side_f]}]
[-core_offset { *value* | *vertical_value horizontal_value* | *side_1 ... side_N*}]
[-row_core_ratio *row_core_ratio*]
[-coincident_boundary]
[-boundary *boundary*]
[-flip_first_row]
[-reset]
[-pin_snap]
[-honor_pad_limit]
[-tile *tile_name*]
[-site_def *site_def_name*]
[-use_site_array]
[-use_site_row]

Data Types

utilization float
row_core_ratio float
boundary vector of coordinates
tile_name string
site_def_name string

ARGUMENTS

-control_type core | die

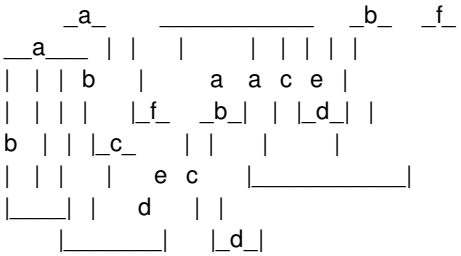
Specifies whether the side_length and side_ratio options apply to the core or the die boundary. If set to die, then the dimensions (side_length and side_ratio) are applied to the die boundary and the core_offset values are subtracted from the dimensions to determine the core boundary. If set to core (default), the dimensions are applied to the core boundary and the core_offset values are added to the dimensions to determine the final die boundary. By default, the control_type is core.

-core_utilization *utilization*

Specifies the utilization of the core area. The utilization is the total area of the core occupied by all standard cells and macro cells divided by the total core area. You can specify a value between 0 and 1. The cell area includes all standard and macro cells. For example, a core utilization of 0.8 specifies that 80 percent of the core area is used for cell placement at this stage. Later, the tool might add more cell area, with the remaining area available for routing. By default, the core utilization is 0.7.

-shape R | L | T | U

Specifies the shape used by the command. If the control_type is die, then the shape applies to the die boundary shape settings. Specifies a template shape used to determine the cell boundary and core shape of the rectilinear block. The following diagram shows the definition of the edges and the orientation of the R-, L-, T-, and U- rectilinear blocks. By default, the core shape is R (rectangular).



-orientation N | W | S | E

Specifies one of four possible orientations for the specified rectilinear shape. The orientations are North (N), West (W), South (S), and East (E). The tool repositions the block to the specified orientation by rotating it in a clockwise direction. For **-shape R**, the orientation is always N.

-side_ratio { *side_a side_b side_c side_d side_e side_f*}

Specifies the side ratio used by the command. If the control_type is die, then it applies the side_ratio to the die boundary side settings. Each dimension in the list represents the relative proportion of the dimension of the edge to the sum of all the dimensions listed. For example, if the list of dimensions of an L-shaped block is {1 2 1 1}, the tool calculates the dimension of side a, c, or d (where the value is 1) as 20% (1/(1+2+1+1)) of the sum of the dimensions listed, and the dimension of side b is 40% of the summation.

-side_length { *side_a side_b side_c side_d side_e side_f*}

Specifies the side length used by the command. If the control_type is die, then it applies to the die boundary side settings. Each dimension in the side_length list represents the length of the edge. If you provide more values than required to describe the specified shape, the extra values are ignored. If you do not provide all of the values required to describe the specified shape, the tool issues an error message. There are only two dimensions for **-shape Rect**: width and height. This option is mutually exclusive with the **-side_ratio** option.

-core_offset { *value* | *vertical_value horizontal_value* | *side_1 ... side_N*}

Specifies the distance between the side of the core and the side of the die boundary. If only one value is specified, the value is used for all sides. If two values are specified, the first value is applied to all vertical edges and the second value is applied to all horizontal edges. Side numbers are based on the standard rectilinear numbering and do not correlate to the numbering scheme used to define the size of each edge (side_a, side_b, etc). By default, the core offset is equal to the minimum I/O cell height. If there are no I/O cells, the core offset is 0.

-row_core_ratio *row_core_ratio*

Specifies the amount of channel area between cell rows in the core area to reserve for routing. The *ratio* is a number between 0 and 1.0. A smaller row-to-core ratio creates more space for routing channels. A value of 1.0 creates no routing channel space. By default, the ratio is 1.0. Note that this ratio should be greater than or equal to the core utilization value.

-coincident_boundary

Uses the existing die boundary. If this option is specified and the core-based constraints result in a core that is too large to fit in

the existing die boundary, the command issues an error message. No default is specified.

-boundary *boundary*

Specifies the shape to be used by the command. If the control_type is core, then the boundary defines the core area and the core_offsets should be added to create the die. If control_type is die, then the core_offset should be subtracted from the die boundary to create the core area. The format is { {x1 y1} {x2 y2} {x3 y3} {x4 y4} }.

-flip_first_row

Specifies whether the command flips the first row at the bottom of the core area for horizontally-placed cell rows, or flips the leftmost row for vertically-placed cell rows. By default, the tool flips the first row at the bottom of the core area.

-reset

Reset all the constraints to their defaults.

-pin_snap

Specifies the snapping of terminal centers to the center of wire tracks. The **-pin_snap** option snaps the center of terminals to the center of the closest wire track for existing pins. Note that this option does not snap power terminals, ground terminals, or terminals with I/O constraints. By default, the setting is off.

-honor_pad_limit

Adjusts the core and die size to honor pad-limited designs. If this option is not specified, the core area is created based on the default core utilization ratio 0.7.

-tile *tile_name*

Specifies the tile name to be used in floorplanning when there are multiple tiles in the technology file.

-site_def *site_def_name*

Specifies the site def to be used in floorplanning when there are multiple site defs in the technology file. The default is to use default site def. If there is no default site def, the command uses the site def with the smallest site width.

-use_site_array

Specifies that the tool creates site arrays.

-use_site_row

Specifies that the tool creates site rows.

DESCRIPTION

Specifies the constraints used to create a floorplan with a boundary, core, site array (or rows), and wire tracks. Before executing this command, you must open a physical design by using the **open_block** command, or create a design with the **read_verilog** or **read_verilog_outline** commands.

EXAMPLES

The following example sets the constraint of utilization to be 0.8.

```
prompt> set_auto_floorplan_constraints -utilization 0.8
1
```

The following example sets the preferred core shape to be a rectangle (R).

```
prompt> set_auto_floorplan_constraints -shape R
1
```

The following example sets the preferred core length to create the floorplan.

```
prompt> set_auto_floorplan_constraints -side_length {200 200}
1
```

SEE ALSO

report_auto_floorplan_constraints(2)
create_io_ring(2)
remove_io_rings(2)
report_io_rings(2)

set_base_lib

Updates the base library location on a sparse library.

SYNTAX

set_base_lib
[-library *library_name*]
-base_lib *base_library_path*

Data Types

library_name string
base_library_path string

ARGUMENTS

-library *library_name*

The sparse library to update the base library path. If no library is specified, the *current_lib* is used. Note that the given library must be a sparse library. Using this command on a non-sparse library will cause an error. Use the *is_sparse* attribute on a library to verify if it is a sparse library.

-base_lib *base_library_path*

A base library path to set on this library. Base library path can be simple, relative, or absolute. A simple library path will be resolved with the first library found on the *search_path*. A note of caution: only design-type libraries can be used as base library, and relying on *search_path* to resolve to a specific library, when there are multiple matches on the *search_path*, the order in the *search_path* decides the library used as base. Note that the base library itself can also be a sparse library, and this command triggers an auto re-synchronizing of the sparse base library with the base libraries.

DESCRIPTION

The **set_base_lib** command modifies the base library location stored on a sparse library. It is useful for keeping the sparse library's reference base library up to date, after the base library has been moved or renamed. All blocks of the sparse library must be closed when this command is used. The library catalog of the updated base library is read, and merged into the sparse library. This merging does not change the local blocks already saved in the sparse library, even if their base library counterparts have been modified, or even deleted. If there are new blocks in the base library, the block entries are appended to the sparse library's catalog.

The sparse library's ref_lib list will also be reset with that of the base library.

The base library and its location can be retrieved with the *base_lib* and *base_lib_path* attribute on the library.

EXAMPLES

The following example sets the base library on the sparse library top_sparse_lib.

```
prompt> set_base_lib -library top_sparse_lib -base_lib /home/user/top_lib
Information: Rebasing Sparse View library 'top_sparse_lib' to base
library '/home/user/top_lib'. (NDM-103)
prompt> get_attribute [get_libs top_sparse_lib] base_lib
{top_lib}
prompt> get_attribute [get_libs top_sparse_lib] base_lib_path
/home/user/top_lib/lib.ndm
```

SEE ALSO

create_lib(2)
current_lib(2)
get_attribute(2)
set_ref_libs(2)
search_path(3)

set_blackbox_clock_port

Specifies the clock ports for black box timing.

SYNTAX

status **set_blackbox_clock_port**
ports

Data Types

ports collection

ARGUMENTS

ports

Specifies the port objects that are clock ports as part of the black box timing information for the current block.

DESCRIPTION

This command specifies the ports that are clock ports of the current block, either clock input ports or clock output ports.

The information is part of the black box timing information. The **commit_blackbox_timing** command uses this information to create and save black box timing data for the current block.

EXAMPLES

The following example specifies port *Clk* as the clock port and specifies the clock network delay from this port.

```
prompt> set_blackbox_clock_port Clk
prompt> create_blackbox_clock_network_delay -value 0.5 [get_ports Clk]
```

SEE ALSO

create_blackbox_clock_network_delay(2)
create_blackbox_constraint(2)
create_blackbox_delay(2)
commit_blackbox_timing(2)

set_blackbox_port_drive

Specifies the driver of ports in black box timing.

SYNTAX

status **set_blackbox_port_drive**
-type *drive_type_name*
[-input_transition_rise *rise_transition_time*]
[-input_transition_fall *fall_transition_time*]
ports

Data Types

drive_type_name string
rise_transition_time float
fall_transition_time float

ARGUMENTS

-type *drive_type_name*

Specifies the drive type for the ports. The *drive_type_name* is the name created by using the *create_blackbox_drive_type drive_type_name* to define the output drive type.

-input_transition_rise *transition_time*

Specifies the rise transition time at the input pin of the driver cell for the given ports. By default, the input rise transition time of the drive type is used.

-input_transition_fall *transition_time*

Specifies the fall transition time at the input pin of the driver cell for the given ports. By default, the input fall transition time of the drive type is used.

ports

Specifies the port objects for which to apply the driver description.

DESCRIPTION

This command defines the driver of a specified set of output or inout ports as part of the black box timing information for the current block.

EXAMPLES

The following example defines the driver at output port *DataOut* as a cell whose reference library cell is *BUF8* in reference library *stdreflib1* with 0.03ns rise and fall transition time at the input pin.

```
prompt> create_blackbox_drive_type -lib_cell [get_lib_cells stdreflib1/BUF8] \
    -input_transition_rise 0.01 -input_transition_fall 0.01 driveType1
prompt> set_blackbox_port_drive -type driveType1 -input_transition_rise 0.03 \
    -input_transition_fall 0.03 [get_ports DataOut]
```

SEE ALSO

create_blackbox_drive_type(2)
create_blackbox_load_type(2)
commit_blackbox_timing(2)
set_blackbox_port_load(2)

set_blackbox_port_load

Sets the load for the specified ports in black box timing.

SYNTAX

status **set_blackbox_port_load**
-value *capacitance_value* | -type *load_type_name*
[-factor *number*]
ports

Data Types

capacitance_value float
load_type_name string
number integer

ARGUMENTS

-value *capacitance_value*

Specifies the load capacitance for the ports. This option is mutually exclusive with the **-type** option.

-type *load_type_name*

Specifies the load type for the ports. The *load_type_name* is the name created with the **create_blackbox_load_type load_type_name** command to define the output load type. This option is mutually exclusive with the **-value** option.

-factor *number*

Specifies the multiplication factor for the load type at the specified ports. You must specify the **-type** option together with the **-factor** option.

ports

Specifies the port objects for which to apply the load description.

DESCRIPTION

This command defines the load of a specified set of input or inout ports as part of the black box timing information for the current block.

EXAMPLES

The following example defines the load at input port *DataIn* as five *BUF8* cells in reference library *stdreflib1*.

```
prompt> create_blackbox_load_type -lib_cell [get_lib_cells stdreflib1/BUF8] loadType1
prompt> set_blackbox_port_load -type loadType1 -factor 5 [get_ports DataIn]
```

The following example defines the load at input port *DataIn* as 0.5pF.

```
prompt> set_blackbox_port_load -value 0.5 [get_ports DataIn]
```

SEE ALSO

create_blackbox_drive_type(2)
create_blackbox_load_type(2)
commit_blackbox_timing(2)
set_blackbox_port_drive(2)

set_block_boundary

Sets the boundary for block.

SYNTAX

int **set_block_boundary**
-cell *block_instance_name*
[-boundary *polyrect*]
[-origin {*x y*}]
[-orientation *orientation*]

Data Types

block_instance_name string
polyrect list
x float
y float
orientation orientation in DEF syntax

ARGUMENTS

-cell *block_instance_name*

Specifies the block instance name.

-boundary *polyrect*

Specifies *polyrect* region as the block instance boundary in current design. The *polyrect* is described as a list of location coordinates.

-origin {*x y*}

Specifies the origin point of the specified block instance. If this option is not specified, the command automatically derives the origin co-ordinates based on the orientation of the block instance. For example: Block instance which has R0 orientation will have the lower-left point of the block instance boundary bounding box as it's origin, block instance with MX orientation will have upper-left point of the block instance boundary bounding box as it's origin and so on.

-orientation orientation

Specifies the orientation of the block instance.

DESCRIPTION

Generate the specified boundary for the block instance.

EXAMPLES

The following example sets the block instance boundary.

```
prompt> set_block_boundary -cell uSUB2 -boundary {{50 50} {300 50} {300 200} {150 200} {150 300} {50 300}}
```

SEE ALSO

explore_logic_hierarchy(2)
block_attributes(3)

set_block_grid_references

Creates a block grid for a block design reference.

SYNTAX

status **set_block_grid_references**
[-designs *design_list*]
[-grid *grid*]
[-snap_point {*x y*}]
[-adjust_snap_point_of_cell *block_instance*]
[-reset]

Data Types

design_list collection
grid block_grid_object
x float
y float
block_instance block_cell_object

ARGUMENTS

-designs *design_list*

Specifies the block reference designs which to setup with block grid information. One block reference design can only be associated with one block grid.

-grid *grid*

Specifies the block grid to associate with the block reference designs. One block grid can be associated with multiple block reference designs.

-snap_point {*x y*}

Specifies the block grid snap point of the block reference designs. A snap point in a block reference design is the point on the block that should be on the associated block grid. By default, the snap point is the origin of the block.

The **shape_blocks** command can move the snap point away from the origin of the block, depending on the needs of block placement. You can use this command and option to change the snap point. The user- specified snap point of a block reference design is fixed.

The **shape_blocks** command honors and does not move the snap point on the block reference if it is fixed. If you did not specify the snap point, it is not fixed. Use **set_block_grid_references -reset** to unfix the snap point and reset it to the origin of the block.

-adjust_snap_point_of_cell *block_instance*

Specifies the block cell instance on which to adjust the snap point. The tool uses the given block cell instance and its current location and orientation to adjust the snap point of the block reference design. The snap point is moved to a location on the block reference design, such that the given block cell instance is on the associated block grid. If there are other cell instances of the same reference design, you can use **snap_cells_to_block_grid** to move those cells on grid with the new snap point location.

-reset

Removes the association between the block grid and the block reference design. Also moves the snap point of the reference design to its origin and unfixes it.

DESCRIPTION

This command sets or resets the association between block reference designs and block grids. The command also sets up or resets snap point information of a block reference design.

EXAMPLES

The following example associates block grid gr1 with block reference BLK1, then changes the associated grid to gr2 and set the snap point of BLK1 to {0 10}.

```
prompt> set_block_grid_references -designs BLK1 -grid gr1
prompt> set_block_grid_references -designs BLK1 -grid gr2 -snap_point {0 10}
```

The following example associates block grid gr1 with block reference BLK2, then set the snap point of BLK2 to {0 10}.

```
prompt> set_block_grid_references -designs BLK2 -grid gr1
prompt> set_block_grid_references -designs BLK2 -snap_point {0 10}
```

SEE ALSO

create_grid(2)
get_grids(2)
remove_grids(2)
report_grids(2)
set_grid(2)
snap_cells_to_block_grid(2)

set_block_pin_constraints

Sets pin constraints on the pins of block cells.

SYNTAX

int **set_block_pin_constraints**
[-allowed_layers *metal_layers*]
[-corner_keepout_num_tracks *tracks*]
[-corner_keepout_distance *distance*]
[-pin_spacing *spacing*]
[-pin_spacing_distance *distance*]
[-sides *side_numbers*]
[-exclude_sides *exclude_sides_numbers*]
[-allow_feedthroughs true | false]
[-stacking_allowed any | none | with_pg_pins_only | with_signal_pins_only]
[-cells *cell_collection*]
[-width *pin_width*]
[-length *pin_length*]
[-self]
[-hard_constraints layer | spacing | location | length | off]

Data Types

metal_layers list
tracks integer
distance real
spacing real
side_numbers list of integers
exclude_sides_numbers list of integers
cell_collection list
pin_width real or list of layer-real pairs
pin_length real or list of layer-real pairs

ARGUMENTS

-allowed_layers *metal_layers*

Specifies the set of metal layers allowed for pin placement. The argument is a list of metal layers.

By default, the metal layers from Min-routing layer to the maximum metal layer (specified earlier during floorplan) are allowed for pin placement. If the maximum metal layer has not been specified, the maximum available metal layer specified in the technology file is used as the default. If the maximum layer specified is beyond the maximum metal layer for the current design, then the maximum metal layer for the current design is used.

-corner_keepout_num_tracks *tracks*

Specifies the distance in wire tracks from the corners of blocks where pin placement should begin.

If neither the **-corner_keepout_num_tracks** nor the **-corner_keepout_distance** option is specified, the default distance is five wire tracks. The value for *tracks* must be zero or a positive integer. This option is mutually exclusive with option "-corner_keepout_distance". Setting this option will invalidate "-corner_keepout_distance" option.

-corner_keepout_distance *corner_keepout_distance*

Specifies the distance in microns from the corners of blocks where pin placement should begin. The distance has to be 0 or a positive integer. By default, the default distance is five wire tracks. This option is mutually exclusive with the **-corner_keepout_num_tracks** option. Setting this option will invalidate **-corner_keepout_num_tracks** option.

-pin_spacing *int*

Specifies the minimum number of wire tracks between adjacent pins or between a pin and a preroute wire that cuts across a block edge in the normal direction. Pins that are created are always snapped to wire tracks. The default pin-to-pin spacing is one wire track. The spacing number must be zero or a positive integer.

-pin_spacing_distance *spacing_distance*

Specifies the minimum distance between adjacent pins or between a pin and a preroute wire that cuts across a block edge in the normal direction. Pins that are created are always snapped to wire tracks.

-sides *side_numbers*

Specifies the block sides on which the pin must be placed.

A side number is a positive integer that starts at one, and increments by one as you proceed clockwise around each edge in the shape. Given any rectilinear shape, the leftmost vertical edge is the starting edge (side number 1). If there are multiple leftmost edges, then the edge 1 is the lowest leftmost edge. You cannot specify a value of 0 for this option.

Side number is counted as observed inside the corresponding reference block itself. You can reset the side number to default value by specifying the **-sides** option with an empty string. For example, to reset the side number, enter

-sides {}

This option is mutually exclusive with the **-exclude_sides** option.

-exclude_sides *exclude_side_numbers*

Specifies the block edges on which pins cannot be placed. The *exclude_side_numbers* argument is a list of positive integers. A side number is a positive integer that starts at one, and increments by one as you proceed clockwise around each edge in the shape. Given any rectilinear shape, the leftmost vertical edge is the starting edge (side number 1). If there are multiple leftmost edges, then the edge 1 is the lowest leftmost edge.

For example, to exclude sides 2, 4, and 6, enter

-exclude_sides {2 4 6}

You can reset the excluded side number to default value by specifying the **-exclude_sides** option with an empty string. For example, to reset the excluded sides, enter

-exclude_sides {}

This option is mutually exclusive with **-sides** option.

-allow_feedthroughs *true* | *false*

Specifies whether feedthrough ports can be created in pin placement flow. By default, this option is *false* and new feedthroughs cannot be created. When feedthroughs are allowed, global routing creates feedthrough routing on blocks as needed. Each top-

level net for which a feedthrough port is created is split into a set of new top-level nets and child-level nets, if feedthrough nets are created for the child nets. Directions are assigned for the newly created feedthrough ports. Because new ports are created in the block cell, the netlist is modified as well.

-stacking_allowed any | none | with_pg_pins_only | with_signal_pins_only

Specifies how signal pins can stack with other pins during pin placement. This option takes one of the four allowed values:

- **any**
  Signal pins can stack with other signal pins on different layers. Signal pins can also stack with power or ground straps, power or ground pins on different layers as well. This is the default value.
- **none**
  Signal pins cannot stack with other signal pins on different layers. Signal pins cannot stack with power or ground traps, power or ground pins on different layers neither.
- **with_pg_pins_only**
  Signal pins cannot stack with other signal pins on different layers. But signal pins can stack with power or ground straps, power or ground pins on different layers.
- **with_signal_pins_only**
  Signal pins can stack with other signal pins on different layers. But signal pins cannot stack with power or ground straps, power or ground pins on different layers.

-cells *cell_collection*

If this option is specified, the constraints are only applied to pins on those blocks in the *cell_collection*. By default, the constraints are applied to pins on all blocks.

-width *pin_width*

Specifies the width of the pin. The width is perpendicular to the layer's preferred routing direction. The syntax can be a single real number to indicate the width is applied to all allowed layers (referred to as common pin width). Or alternatively, the width can be specified as a list of layer-real pairs as the following (referred to as per layer pin width):

-width {{M2 0.2} {M3 0.2} {M4 0.3} {M5 0.3}}

It means that if the pin is to be placed on layer M2 or M3, its width should be 0.2 and if on M4 or M5, 0.3. On other layers, use the default width defined by the tech file.

If user first sets the pin width to be common pin width (or per layer pin width), then in a second **set_block_pin_constraints** sets the pin width to be per layer width (or common pin width), then the width specified in the second command will override the first.

However, if user first sets the pin width to be per layer, then in a second command sets the pin width again to be per layer but with different layers or different values, then the combined per layer pin width will be the final constraints. For example, the following example:

```
set_block_pin_constraints -cells mid1 -width 0.3
set_block_pin_constraints -cells mid1 -width {{M2 0.3} {M3 0.3}}
```

The first common pin width constraint will be overridden by the second per layer pin width. The pin width should be 0.3 on layer M2 or M3, but default width on all other layers.

And for the following example:

```
set_block_pin_constraints -cells mid1 -width {{M3 0.4} {M4 0.4}}
set_block_pin_constraints -cells mid1 -width {{M2 0.3} {M3 0.3}}
```

The pin width should be 0.3 on layer M2 or M3, 0.4 on layer M4, and default width on all other layers.

-length *pin_length*

Specifies the length of the pin. The length is in parallel to the layer's preferred routing direction. The syntax can be a single real number to indicate the length is applied to all allowed layers (referred to as common pin length). Or alternatively, the length can be specified as a list of layer-real pairs as the following (referred to as per layer pin length):

-length {{M2 0.2} {M3 0.2} {M4 0.3} {M5 0.3}}

It means that if the pin is to be placed on layer M2 or M3, its length should be 0.2 and if on M4 or M5, 0.3. On unspecified layers, the pin length is derived from the pin width. Refer to the manpage of **place_pins** for details.

-self

Applies constraints only to the top-level design. By default, the constraints are applied to block cells inside current top design.

-hard_constraints off | spacing | location | layer | length

Specifies hard constraint types. The supported hard constraint types are spacing, location, layer, or length. If none of the constraints should be hard, user can use the keyword off.

When one particular type of constraint is specified as hard, **place_pins** honors it without any design rule checks, even if honoring it means one or more design rule violations.

If spacing is specified as hard, then the minimum spacing between applicable pins is at least the value specified by the associated spacing constraints.

If location is specified as hard, then the applicable pin is placed at the exact side and offset specified by the associated pin constraints. If the location constraint is specified as coordinates by **set_individual_pin_constraints** instead, the coordinates are snapped to the edge unless option **-off_edge** is presented, then during **place_pins** the pin is placed at the snapped edge and offset (option **-off_edge** is not presented) or the specified coordinates (option **-off_edge** is presented).

If layer is specified as hard, then the applicable pin is placed at the layer specified by the associated layer constraints.

If length is hard, then the applicable pin's length is exactly the value specified by the associated length constraints.

If two constraints conflict and only one of those two constraints is specified as a hard constraint, the other constraint is ignored. For example, you can specify a layer constraint and a side constraint on a pin. When the layer's preferred direction is horizontal and the side is horizontal edge, if the layer constraint set to hard constraint, **place_pins** places pins on the horizontal layer and vertical edge as there is no legal horizontal wiretrack on horizontal edge.

You can specify more than one hard constraint type.

By default, this option is off.

DESCRIPTION

This command sets pin placement constraints on blocks. The constraints are honored during pin placement and are saved in the design database.

The **set_block_pin_constraints** command is additive. The constraint values set by previous **set_block_pin_constraints** commands are retained unless they are explicitly overridden by subsequent calls

The **set_editability** command can enable or disable the **set_block_pin_constraints** command for specified blocks or levels of physical hierarchy. This command has no effect on the blocks that are disabled with the **set_editability** command.

This command returns 1 on success, 0 otherwise.

EXAMPLES

Following example sets the allowed block pin layers to METAL2 and METAL3, and allows feedthroughs on all the blocks in the design.

```
prompt> set_block_pin_constraints -allowed_layers [get_layers METAL2 METAL3] \
  -allow_feedthroughs true
```

SEE ALSO

place_pins(2)
remove_block_pin_constraints(2)
report_block_pin_constraints(2)
set_individual_pin_constraints(2)
set_editability(2)

set_block_to_top_map

Specifies the relationship between the top-level mode, corner, and clock objects to objects inside of instances of lower level designs.

SYNTAX

int **set_block_to_top_map**
-block *cell* | -path *cell_name* |
-auto_clock connected | local | all
[-mode {*block_mode top_mode*}]
[-corner {*block_corner top_corner*}]
[-clock {*top_mode block_clock top_clock*}]
[-inverted_clock {*top_mode block_clock top_clock*}]
[-unused_clock {*top_mode block_clock*}]
[-ignore_waveform]
[-report_only]

Data Types

cell string
cell_name string
block_mode string
top_mode string
block_corner string
top_corner string
block_clock string
top_clock string

ARGUMENTS

-block *cell*

Specifies a cell in the current design in which to map objects. The cell must be an instance of a separate design in the hierarchy. That is, the cell must not be a leaf cell or simply a logic hierarchy. You must specify one of the **-block**, **-path** or **-auto_clock** options.

-path *cell_name*

Specifies the name of a cell in which to map objects. The cell must be an instance of a separate design in the hierarchy. That is, the cell must not be a leaf cell or simply a logic hierarchy. It is not necessary that the cell currently be in memory, however, the cell name, block modes, block corners, and block clocks will not be checked for errors. If the cell name or block object is not eventually read into memory, the setting associated with **-path** will not be used. You must specify either **-path**, **-block** or **-auto_clock**. You should use **-block** instead of **-path** for cells in memory so that error checking is performed.

-auto_clock connected | local | all

Automatically derives any clock associations between top-level clocks and block-level clocks. Clocks are mapped for all blocks and modes. You can use this option to avoid specifying multiple **-clock** options. In order for **-auto_clock** to work properly, you must first use the **-mode** option to define any mode mappings where the block and top mode names do not match. The command traces the clocks and determines where top-level clocks cross into the blocks to build the list of associations. The **-auto_clock** function will not change clock associations that were set with the **-clock**, **-unused_clock**, or with previous **set_block_to_top map -auto_clock** commands.

You can select one of three rules for auto-clock mapping:

- **connected** - If you select the "connected" rule, block-level clocks will be associated with top-level clocks under the following conditions: If a block clock has its source at a block port (a boundary clock), it can be associated to a top-level clock that propagates to that port. If a block clock has its source on an internal pin sources (a local clock), the top-level clock source must be declared at the same pin.
- **local** - If you select the "local" rule, connected clocks will be mapped. In addition, if a block clock has its source on an internal pin sources (a local clock) and has no associated top-level clock, a new top-level clock will be created with the same waveform as the block-level clock. The new top-level clock will be associated with the local block-level clock.
- **all** - If you select the "all" rule, clocks will be mapped as in the "local" rule. In addition, if a block clock has its source at a block port (a boundary clock) and no top-level clock propagates to that port, a new top-level clock will be created with the same waveform as the block-level clock. The new top-level clock will be associated with the local block-level clock. The top-level boundary clock will allow timing analysis to be done, but it is not necessarily correct for signoff. You should consider removing or replacing this clock before signoff.

Note that if two block-level clocks are declared at the same boundary port, the "all" rule cannot resolve both of them. This is because it will not automatically declare two conflicting clocks in the top-level block. You must manually declare two top-level clocks, with the appropriate waveforms. After the top clocks exist, **-auto_clock** mapping can associate them with the block clocks.

-mode

Specifies the mapping between mode constraints in a block and mode constraints at the top level. For example, you can associate constraints from block mode "A" with top mode "B". You can specify multiple **-mode** options in the same **set_block_to_top_map** command, and specify all mode mappings in one command. If you don't specify a mapping for a block mode, the command assumes that the block mode and the top mode have the same name.

-corner

Specifies a mapping between corner constraints in a block and corner constraints at the top level. For example, you can associate constraints from block corner "A" with top corner "B". You can specify multiple **-corner** options in one **set_block_to_top_map** command to specify all corner mappings for a block at one time. If you don't specify a mapping for a block corner, it will be assumed that the block corner and the top corner have the same name.

-clock

Specifies a mapping between clock objects in a block and clock objects at the top level. For example, you can associate constraints from block clock "CLK" with top clock "SYS_CLK". You can specify multiple **-clock** options with one **set_block_to_top_map** command to specify all clock mappings for a block at one time. When specifying a clock mapping, you must also state the mode associated with the top-level clock. This helps disambiguate between top-level clocks that have the same name in different modes. You can use the **-auto_clock** option to have the tool to generate clock mappings automatically.

-inverted_clock

Inverts the waveform of the clock in the block from the waveform at the top level. Use this option when there is an inverter between the top-level clock source and the block boundary.

-unused_clock

Specifies that a clock defined at the block level should be ignored and should not to be used to constrain the top-level design. You can specify multiple **-unused_clock** options with one **set_block_to_top_map** command to specify all unused clocks for a block at once. When specifying an unused clock mapping you must also specify the top-level mode in which the clock will be

unused.

-ignore_waveform

By default, you cannot associate a block-level clock with a top-level clock unless the clock waveforms match exactly. The **-ignore_waveform** option causes the **-clock** and **-auto_clock** options to allow associations, even though the clock waveforms don't match.

-report_only

If you use the option along with **-auto_clock**, then the tool prints out all the automatic clock mappings that it is able to derive. However, only the association is printed; the association is not changed in the database. This allows you to preview the mapping results before committing to them.

DESCRIPTION

Specifies the relationship between top-level mode, corner, and clock objects to objects inside of instances of lower level designs. These associations are used by commands such as **write_io_constraints**.

For example, you can specify that top-level mode "mission" is associated with block-level mode "func". When you want to create budgets for the various blocks at the top level, the top level will be timed in "mission" mode. But when constraints are written for the block, they will be written for "func" mode. It is possible that many top-level modes might be associated with the same block level mode. Or, some block modes might not be used for top-level timing.

Mappings for corners and clocks can also be provided. You can also specify that some block-level clocks are not to be considered for top-level timing.

It can be tedious to provide a block-to-top mapping for all objects. By default, mode and corners are automatically associated if they have the same name. For clocks, you can use the **-auto_clock** option to let the tool derive clock mappings for you. Manual specification of clock mappings might be needed in case of ambiguity or error.

Block-to-top mappings can be written out using the **write_script** command, so they can be easily reapplied to new netlists. You can view the current mappings by using the **report_block_to_top_map** command.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following example sets mode, corner, and clock mappings for block core/CPU. Block mode "func" is used along with top-mode bscan. The other modes and corners are mapped based on matching names. The clock CLK inside of the block has different top-level associations for each top-level mode. For top-level mode "func", the clock "CLK2" inside of the block is not used:

```
prompt> set_block_to_top_map -block core/CPU -mode {test bscan} \
    -clock {func CLK SYS_CLK} -clock {test CLK TCLK} \
    -clock {bscan CLK TCLK} -unused_clock {func CLK2}
```

The following automatically associates block and top clocks which are connected in your design:

```
prompt> set_block_to_top_map -auto_clock connected
```

The following automatically associates block and top clocks which are connected in your design, and creates new top-level clocks where connections cannot be found.

```
prompt> set_block_to_top_map -auto_clock all
```

SEE ALSO

report_block_to_top_map(2)
report_clocks(2)
report_corners(2)
report_modes(2)
write_io_constraints(2)
write_script(2)

set_boundary_budget_constraints

Creates or modifies constraints for block boundaries of budgeted blocks.

SYNTAX

int **set_boundary_budget_constraints**
-name *boundary_name*
[-driving_cell *lib_cell_name*]
[-library *lib_name*]
[-pin *pin_name*]
[-from_pin *from_pin_name*]
[-input_transition_rise *rtrans*]
[-input_transition_fall *ftrans*]
[-fanin_capacitance *capacitance*]
[-load_capacitance *capacitance*]
[-max_transition *transition*]
[-corner *corner_name* | -default]
[-auto]

Data Types

boundary_name string
lib_cell_name string
lib_name string
pin_name string
from_pin_name string
rtrans float
ftrans float
capacitance float
transition float
corner_name string

ARGUMENTS

-name *boundary_name*

Specifies the name of the boundary being defined. You can use this name along with the **-early_boundary** and **-late_boundary** options of the **set_pin_budget_constraints** command to associate constraints with a particular budgeted pin.

-driving_cell *lib_cell_name*

Specifies the name of a driving cell to drive a budgeted input pin. This option creates a **set_driving_cell** constraint in your final budget.

-library *lib_name*

Specifies the library of your driving cell. If you don't specify the library, your driving cell will be automatically selected from your reference libraries.

-pin ***pin_name***

Specifies the output pin on the driving cell to use. By default, the first output of the driving cell is used.

-from_pin ***from_pin_name***

Specifies which input pin of your driving cell is used when evaluating the drive and delay of the output pin. By default, the pin resulting in the worst drive is used.

-input_transition_rise ***rtrans***

Specifies the rising transition to be used at the input pin of your driving cell. By default, zero transition is used.

-input_transition_fall ***ftrans***

Specifies the falling transition to be used at the input pin of your driving cell. By default, zero transition is used.

-fanin_capacitance ***capacitance***

Specifies the **set_load** value to apply to input pins in your budget. This option must be accompanied either by **-corner** or **-default**.

-load_capacitance ***capacitance***

Specifies the **set_load** value to apply to output pins in your budget. This option must be accompanied either by **-corner** or **-default**.

-max_transition ***transition***

Specifies the **max_transition** value to apply to boundary pins in your budget. This option must be accompanied either by **-corner** or **-default**.

-corner ***corner_name*** | **-default**

Specifies that values given for the **-fanin_capacitance** and **-load_capacitance** options should be applied in the process corner with the given name.

-default

Specify that values given for the **-fanin_capacitance** and **-load_capacitance** options should be applied in process corners do not have specific values given with the **-corner** option. If no specific **-corner** value and no **-default** value is given, zero is used.

-auto

This option is set by the tool when a boundary constraint was automatically derived. If a new automatic constraint is calculated in the future, the current automatic constraint will be deleted. User boundary constraints are never deleted by the tool.

DESCRIPTION

Creates or modifies constraints for block boundaries of budgeted blocks. Use this command to create a "budget boundary" definition that specifies how to apply driving cell and loads to pins of budgeted blocks. You can call **set_boundary_budget_constraints** multiple times with the same boundary name to modify or augment your boundary definition. The boundary definition can be associated with specific budgeted pins by using the **-early_boundary** and **-late_boundary** options of the **set_pin_budget_constraints** command.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following declares a driving cell that should is applied to the input pins of "block1". An additional 0.1 of capacitance is also applied to the input:

```
prompt> set_boundary_budget_constraints -name DCbuf8 -driving_cell bufX8
prompt> set_boundary_budget_constraints -name DCbuf8 -default -fanin_capacitance 0.1
prompt> set_pin_budget_constraints -late_boundary DCbuf8 -input block1/*
```

The following sets output loads for all budgeted pins. The value is different for different corners.

```
prompt> set_boundary_budget_constraints -name OUTload -default \
  -load_capacitance 0.1
prompt> set_boundary_budget_constraints -name OUTload -corner c1 \
  -load_capacitance 0.15
prompt> set_boundary_budget_constraints -name OUTload -corner c2 \
  -load_capacitance 0.05
prompt> set_pin_budget_constraints -late_boundary OUTload -output -all
```

SEE ALSO

compute_budget_constraints(2)
report_budget(2)
set_boundary_budget_constraints(2)
set_budget_options(2)
set_latency_budget_constraints(2)
set_pin_budget_constraints(2)
write_budgets(2)

set_boundary_cell

Sets the boundary-cell configuration for the specified ports.

SYNTAX

status **set_boundary_cell**
[-class wrapper]

[-type type]
[-ports port_list]
[-shift_clk *bcell_shift_clk*]
[-instance *hier_cell_list*]
[-exclude *cell_list*]
[-reuse_threshold *threshold_value*]

Data Types

port_list list
type string
hier_cell_list list
cell_list list

ARGUMENTS

-class wrapper

Specifies the name of the class for which the configuration applies. The only valid value is **wrapper**, which is the default.

-type type

Specifies the cell type to be used for the boundary cell in DFT insertion. There is no default value for this option. If this option is not specified, the tool uses the **WC_D1** type for the specified ports.
The following values are valid for *type*: **WC_D1** | **WC_S1** | **none**

-ports port_list

Specifies the list of ports for which the configuration applies. There is no default value for this option. You must specify the **-ports**option.

-shift_clk *bcell_shift_clk*

Specifies the shift clock to be used for the boundary cell.

This option is applicable only with the **-class core_wrapper** and either of the **-type WC_D1** or **-type WC_D1_S** options.

The value of this option must be a port or an internal pin. The specified port or pin is connected to the shift clock pin of the dedicated boundary cell.

The timing information of the specified clock is obtained from DRC.

There is no default value for this option.

-instance *hier_cell_list*

This option is applicable only when the **-class** option is set to **wrapper**.

This option specifies a list of hierarchical cells. If a port is connected to a register inside one of these hierarchical cells, the tool adds the specified type of boundary cell to the port.

The valid types of boundary cells supported with this option are **none** or **WC_D1**.

If the type of the boundary cell is chosen as **none**, no wrapper cell is added to the port.

If the type of the boundary cell is chosen as **WC_D1**, a dedicated wrapper cell is added to the port.

This option is mutually exclusive with the **-port** option.

There is no default value for this option.

-exclude *cell_list*

Applies only when the class option is set to **wrapper**.

This option specifies the list of I/O register cells to be excluded for the specified port.

To specify a list of cells to be excluded from all the ports, do not specify the port names with the command.

There is no default value for this option.

-reuse_threshold *threshold_value*

Specifies the maximum number of I/O registers that can be reused as shared wrapper cells for the specified ports.

This option is applicable only when the **-class** option is set to **wrapper**.

When the number of I/O registers detected for the specified ports exceeds the specified threshold value, a dedicated wrapper cell is added to the ports.

The default value of this option is -1; that is, the reuse threshold value specified with wrapper configuration is applicable to the specified ports.

If the value of the option is set to 0, all I/O registers associated with the specified ports are reused as shared wrapper cells.

DESCRIPTION

This command sets the boundary-cell configuration for the specified ports.

EXAMPLES

```
prompt> set_boundary_cell -ports { i_din[0] o_dout[0] } -type WC_D1
```

SEE ALSO

set_wrapper_configuration(2)

set_boundary_cell_rules

Sets the boundary cell insertion and checking rules.

SYNTAX

status **set_boundary_cell_rules**
[-left_boundary_cell *lib_cell_name*]
[-right_boundary_cell *lib_cell_name*]
[-bottom_boundary_cells *lib_cell_name*]
[-top_boundary_cells *lib_cell_name*]
[-bottom_left_outside_corner_cell *lib_cell_name*]
[-bottom_right_outside_corner_cell *lib_cell_name*]
[-top_left_outside_corner_cell *lib_cell_name*]
[-top_right_outside_corner_cell *lib_cell_name*]
[-bottom_left_inside_corner_cells *lib_cell_name*]
[-bottom_right_inside_corner_cells *lib_cell_name*]
[-top_left_inside_corner_cells *lib_cell_name*]
[-top_right_inside_corner_cells *lib_cell_name*]
[-mirror_left_outside_corner_cell]
[-mirror_right_outside_corner_cell]
[-mirror_left_inside_corner_cell]
[-mirror_right_inside_corner_cell]
[-mirror_left_boundary_cell]
[-mirror_right_boundary_cell]
[-mirror_left_inside_horizontal_abutment_cell]
[-mirror_right_inside_horizontal_abutment_cell]
[-tap_distance *distance*]
[-top_tap_cell *lib_cell_name*]
[-bottom_tap_cell *lib_cell_name*]
[-prefix *boundary_cell_prefix*]
[-separator *boundary_cell_separator*]
[-insert_into_blocks]
[-at_va_boundary]
[-no_1x]
[-min_row_width *width*]
[-min_horizontal_jog *distance*]
[-min_vertical_jog *distance*]
[-min_horizontal_separation *width*]
[-min_vertical_separation *width*]
[-add_metal_cut_allowed]
[-do_not_swap_top_and_bottom_inside_corner_cell]

Data Types

boundary_cell_prefix string
boundary_cell_separator string
lib_cell_name string

orientation	orientation in DEF syntax
distance	float
width	float

ARGUMENTS

-left_boundary_cell *lib_cell_name*

Specifies the library cell to be placed at the beginning of each cell row.

-right_boundary_cell *lib_cell_name*

Specifies the library cell to be placed at the end of each cell row.

-bottom_boundary_cells *lib_cell_name*

Specifies the library cells to be placed along the bottom object boundaries. In a double-back design, adjacent rows are flipped; the bottom boundary cell is used on unflipped bottom boundary rows and on flipped top boundary rows.

-top_boundary_cells *lib_cell_name*

Specifies the library cells to be placed along the top of object boundaries. In a double-back design, adjacent rows are flipped; the top boundary cell is used on unflipped top boundary rows and on flipped bottom boundary rows.

-bottom_left_outside_corner_cell *lib_cell_name*

Specifies the library cell to be placed at bottom-left outside corners. An outside corner is a corner with a 90-degree inside angle. A bottom corner is a corner that is on a bottom boundary. In a double-back design, adjacent rows are flipped; the bottom corner cell is used on unflipped bottom boundary rows and on flipped top boundary rows.

-bottom_right_outside_corner_cell *lib_cell_name*

Specifies the library cell to be placed at bottom-right outside corners. An outside corner is a corner with a 90-degree inside angle. A bottom corner is a corner that is on a bottom boundary. In a double-back design, adjacent rows are flipped; the bottom corner cell is used on unflipped bottom boundary rows and on flipped top boundary rows.

-top_left_outside_corner_cell *lib_cell_name*

Specifies the library cell to be placed at top-left outside corners. An outside corner is a corner with a 90-degree inside angle. A top corner is a corner that is on a top boundary. In a double-back design, adjacent rows are flipped; the top corner cell is used on unflipped top boundary rows and on flipped bottom boundary rows.

-top_right_outside_corner_cell *lib_cell_name*

Specifies the library cell to be placed at top-right outside corners. An outside corner is a corner with a 90-degree inside angle. A top corner is a corner that is on a top boundary. In a double-back design, adjacent rows are flipped; the top corner cell is used on unflipped top boundary rows and on flipped bottom boundary rows.

-bottom_left_inside_corner_cells *lib_cell_name*

Specifies the library cell to be placed at the bottom-left of inside corners. An inside corner is a corner with a 270-degree inside angle. A left inside corner cell is an inside corner cell on the left end of the horizontal edge that makes up the inside corner.

-bottom_right_inside_corner_cells *lib_cell_name*

Specifies the library cell to be placed at the bottom-right of inside corners. An inside corner is a corner with a 270-degree inside angle. A right inside corner cell is an inside corner cell on the right end of the horizontal edge that makes up the inside corner.

-top_left_inside_corner_cells ***lib_cell_name***

Specifies the library cell to be placed at the top-left of inside corners. An inside corner is a corner with a 270-degree inside angle. A left inside corner cell is an inside corner cell on the left end of the horizontal edge that makes up the inside corner.

-top_right_inside_corner_cells ***lib_cell_name***

Specifies the library cell to be placed at the top-right of inside corners. An inside corner is a corner with a 270-degree inside angle. A right inside corner cell is an inside corner cell on the right end of the horizontal edge that makes up the inside corner.

-mirror_left_outside_corner_cell

Places left outside corner cells in a mirrored orientation.

-mirror_right_outside_corner_cell

Places right outside corner cells in a mirrored orientation.

-mirror_left_inside_corner_cell

Places left inside corner cells in a mirrored orientation.

-mirror_right_inside_corner_cell

Places right inside corner cells in a mirrored orientation.

-mirror_left_boundary_cell

Places left boundary cells in a mirrored orientation.

-mirror_right_boundary_cell

Places right boundary cells in a mirrored orientation.

-mirror_left_inside_horizontal_abutment_cell

Places left inside horizontal abutment cells in a mirrored orientation.

-mirror_right_inside_horizontal_abutment_cell

Places right inside horizontal abutment cells in a mirrored orientation.

-tap_distance ***distance***

Specifies the distance in microns between tap cells.

-top_tap_cell ***lib_cell_name***

Specifies the tap cell to be placed at the top boundary. The tool inserts this tap cell at the specified interval on top boundary rows. This tap cell is for the boundary cells and should not be confused with tap cells for standard cells. In a double-back design, adjacent rows are flipped; the top tap cell is used on unflipped top boundary rows and on flipped bottom boundary rows. This option must be used with the **-tap_distance** option.

-bottom_tap_cell ***lib_cell_name***

Specifies the tap cell to be placed at the bottom boundary. The tool inserts this tap cell at the specified interval on bottom boundary rows. This tap cell is for the boundary cells and should not be confused with tap cells for standard cells. In a double-back design, adjacent rows are flipped; the bottom tap cell is used on unflipped bottom boundary rows and on flipped top boundary rows. This option must be used with the **-tap_distance** option.

-prefix ***boundary_cell_prefix***

Specifies the prefix for the created boundary cells. By default, no prefix is added. When you use this option, the command uses

the following naming convention for created boundary cells:

boundarycell!*prefix*!*library_cell_name*!*number*

By default, no prefix is added and the command uses the following naming convention:

boundarycell!*library_cell_name*!*number*

-separator *boundary_cell_separator*

Specifies the separator character that is used when composing the instance name of the boundary cell. The boundary cell instance name consists of a prefix, the library reference cell name and an incrementing number. For example, the instance name "boundarycell!MY_CELL!25" has the boundary cell prefix, the MY_CELL library reference cell name, the incrementing number 25, and the boundary cell separator !. This naming convention allows you to do pattern matching selection of the boundary cells.

By default, the separator character is an exclamation mark (!) and the command uses the following naming convention for created boundary cells:

boundarycell![*prefix*!]*library_cell_name*!*number*

-insert_into_blocks

Performs boundary cell insertion in the subblocks. By default, the tool performs boundary cell insertion only in the top-level block.

-at_va_boundary

Places horizontal boundary cells on both sides of the voltage area boundaries. When you use this option, rows are treated as cut by voltage area boundaries. Therefore, boundary cells are placed on both sides of a voltage area boundary. By default, voltage area boundaries are ignored during boundary cell insertion.

-no_1x

Prevents the tool from placing boundary cells on a row when the row length equals two times the corner cell width plus one unit tile width. Note that if the row length equals two times the corner cell width, the tool inserts boundary cells on that row.

This option can be used only when the **-top_right_outside_corner_cell**, **-bottom_right_outside_corner_cell**, **-top_left_outside_corner_cell** and **-bottom_left_outside_corner_cell** options are all specified.

-min_row_width *width*

Skips rows with a row width that is smaller than the specified width value.

-min_horizontal_jog *distance*

Specify the minimum horizontal jog distance required for boundary cell placement. If the minimum length is not met, placement blockages will be added by **compile_boundary_cells -add_placement_blockage** to either eliminate the jog or lengthen the jog to meet the minimum length in boundary cell placement.

-min_vertical_jog *distance*

Specify the minimum vertical jog distance required for boundary cell placement. If the minimum length is not met, placement blockages will be added by **compile_boundary_cells -add_placement_blockage** to either eliminate the jog or lengthen the jog to meet the minimum length in boundary cell placement.

-min_horizontal_separation *width*

Specify the minimum horizontal separation required for boundary cell placement. A horizontal separation width is the vertical channel width. If the minimum separation is not met, placement blockages will be added by **compile_boundary_cells -add_placement_blockage** to eliminate the offending standard cell area.

-min_vertical_separation *width*

Specify the minimum vertical separation required for boundary cell placement. A vertical separation width is the horizontal channel width. If the minimum separation is not met, placement blockages will be added by **compile_boundary_cells -add_placement_blockage** to eliminate the offending standard cell area.

-add_metal_cut_allowed

Adds metal cut allowed underneath the standard cell placeable area. Metal cut allowed and forbidden preferred grid extension routing guides are created. Metal cut allowed routing guides cover the area taken up by all the placable site rows reduced by the vertical and horizontal shrink factor. The vertical shrink factor is expressed as a percentage of the smallest site row height and the default is 50%. It can be set using the chipfinishing.metal_cut_allowed_vertical_shrink_factor app option. The horizontal shrink factor is expressed as a percentage of the smallest site row width. It can be set using the chipfinishing.metal_cut_allowed_horizontal_shrink_factor app option. Finally, forbidden preferred grid extension routing guides are created to cover the remaining area up to the boundary.

-do_not_swap_top_and_bottom_inside_corner_cell

Prevents swapping of the top and bottom inside-corner cells for flipped rows.

By default, if a top inside-corner cell is on a flipped row, the tool uses a bottom inside-corner cell instead. If a bottom inside-corner cell is on a flipped row, the tool uses a top inside-corner cell instead.

DESCRIPTION

This command sets the boundary cell insertion and checking rules for the current design. You can run this command multiple times to set the boundary cell rules incrementally.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the boundary cell rules for the left and right edges.

```
prompt> set_boundary_cell_rules -left_boundary_cell myLib/CellLeft \
  -right_boundary_cell myLib/CellRight
```

SEE ALSO

remove_boundary_cell_rules(2)
report_boundary_cell_rules(2)
compile_boundary_cells(2)
check_boundary_cells(2)
compile_targeted_boundary_cells(2)
check_targeted_boundary_cells(2)

set_boundary_optimization

Sets the boundary_optimization restriction on specified pins, cells or modules, thus allowing/restricting optimization across hierarchical boundaries.

SYNTAX

int **set_boundary_optimization**
object_list all | none | auto
[-constant_propagation true | false]
[-unloaded_propagation true | false]
[-equal_opposite_propagation true | false]
[-phase_inversion true | false]

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of pins, cells or modules where to enable/disable boundary optimization. Cell or reference names in object_list must be from the current design. If more than one object is specified, they must be enclosed in quotation marks ("") or braces ({}).

all | **auto** | **none**

Specifies the value of the boundary optimization restriction. All: refers to all Boundary optimization types are true constant_propagation - true unloaded_propagation - true equal_opposite_propagation - true phase_inversion - true

auto: refers to phase_inversion and equal_opposite_propagation false and rest true constant_propagation - true unloaded_propagation - true equal_opposite_propagation - false phase_inversion - false

none: refers to all Boundary optimization types are false constant_propagation - false unloaded_propagation - false equal_opposite_propagation - false phase_inversion - false

-constant_propagation true | **false**

Specifies the value of constant propagation restriction on the specified objects. If true (default), constant propagation is allowed.

-unloaded_propagation true | **false**

Specifies the value of unloaded propagation restriction on the specified objects. If true (default), unloaded propagation is allowed.

-equal_opposite_propagation true | **false**

Specifies the value of equal opposite propagation restriction on the specified objects. If true, equal opposite propagation is allowed. The default value is same as the value of the boundary optimization restriction.

-phase_inversion true | false

Specifies the value of phase inversion restriction on the specified objects. If true, phase inversion is allowed. The default value is same as the value of the boundary optimization restriction.

DESCRIPTION

This command sets the boundary_optimization restrictions on object_list. The restriction allows/restricts the pins, cells or modules to be optimized across hierarchical boundaries. The **compile_fusion** command generally performs four kinds of cross boundary optimizations - constant propagation, unloaded propagation, equal opposite propagation and phase inversion. The specific optimizations can be allowed or restricted on the specified objects by using the corresponding optimization options.

Note: Boundary optimization settings over objects follows below specified priority. pin > cell > module. It implies that change in the settings of cell will not affect the settings of pin (if already restricted with boundary optimization settings) because it has higher priority.

Calling boundary_optimization with the value "auto" does not disable constant propagation and unloaded propagation across hierarchical boundaries of the specified object. To disable those propagations also, -constant_propagation false -unloaded_propagation false needs to be used with the command.Calling set_boundary_optimization with the value "all" will enable all BO types while calling set_boundary_optimization with the value "none" will disable all BO types

Occasionally, cells use an input signal only in its complemented form. In this case, it is best to invert the signal and its ports. The set_boundary_optimization command considers these optimizations (phase inversion). When this occurs, port names are changed.

Note that the **compile.optimization.constant_and_unloaded_propagation_with_no_boundary_optimization** application option does not have any effect on this command.

It is to imply that all | none | auto values are mutually exclusive with individual options i.e. -constant_propagation , -unloaded_propataion, -equal_opposite_propagation, -phase_inversion. Demostration is given in example

This command puts a list of read only attributes on the specified objects. The attributes are boundary_optimization, constant_propagation, unloaded_propagation, equal_opposite_propagation and phase_inversion. Note that, boundary_optimization attribute is an umbrella attribute. If any of the other 4 attribute is false, boundary_optimization is set to false. If all other 4 attributes are true, boundary_optimization is set to true.

To remove this restriction, use the remove_boundary_optimization command.

To report existing restrictions, use the report_boundary_optimization command.

EXAMPLES

The following example shows how to ignore hierarchical boundaries during optimization for cells named U0 and U1 and how to preserve them for a cell named U2 and U3 where U2 still allows constant and unloaded propagation.

```
prompt> set_boundary_optimization [get_cells -regexp U0|U1] all

prompt> set_boundary_optimization [get_cells U2] auto

prompt> set_boundary_optimization [get_cells U3] none
```

The following example shows how to disable only phase inversion optimization on pin U4/out whereas all other optimizations are allowed.

```
prompt> set_boundary_optimization [get_pins U4/out] -phase_inversion false
```

The following example shows how to disable equal-opposite propagation and phase inversion optimization on module named mid1 whereas all other optimizations are allowed.

```
prompt> set_boundary_optimization [get_modules mid1] auto
```

The following example shows how it errors out when all | auto | none are set along with individual options. Both the option setting are mutual exclusive and should not be used together.

```
prompt> set_boundary_optimization [get_modules mid1] auto -constant_propagation false
```

The following example shows how set_boundary_optimization setting should be done for global as well as individual options

```
prompt> set_boundary_optimization [get_modules mid1] auto
prompt> set_boundary_optimization [get_modules mid1] -constant_propagation false
```

SEE ALSO

compile_fusion(2)
remove_boundary_optimization(2)
report_boundary_optimization(2)
reset_design(2)
uniquify(2)

set_budget_margins

Define timing margins to be used in budgets

SYNTAX

int **set_budget_margins**
[-launch *delay*]
[-capture *delay*]
[-setup]
[-hold]
[-prects]
[-postcts]
[-target]
[-actual]
[-clock *budget_clock_spec*]
[-from_clock *budget_clock_spec*]
[-to_clock *budget_clock_spec*]
[-corner *corner_name* | -default]

Data Types

delay float
budget_clock_spec string
corner_name string

ARGUMENTS

-launch *delay*

Specify a margin to be added to the launching block part of a budgeted path. The margin will be added value of the **set_clock_uncertainty** statement in the output SDC of the **write_budgets** command. If **-capture** is also specified for the same path, the effective margin for the whole path will be the sum of the two margins.

-capture *delay*

Specify a margin to be added to the capturing block part of a budgeted path. The margin will be added value of the **set_clock_uncertainty** statement in the output SDC of the **write_budgets** command. If **-launch** is also specified for the same path, the effective margin for the whole path will be the sum of the two margins.

-setup

Only apply the margin to setup paths. By default, the margin is applied to both setup and hold paths.

-hold

Only apply the margin to hold paths. By default, the margin is applied to both setup and hold paths.

-prects

Only apply the margin to parts of the path that are in the "prects" budget mode. The budget mode of the path is determined by the **-adjust_latency** option of the **set_budget_options** command. Some blocks may be in the prects mode while others are in postcts modes. By default, the margin is applied in all modes.

-postcts

Only apply the margin to parts of the path that are in one of the postcts budget modes. The budget mode of the path is determined by the **-adjust_latency** option of the **set_budget_options** command. Both "target" and "actual" modes are considered to be postcts. Some blocks may be in the prects mode while others are in postcts modes. By default, the margin is applied in all modes.

-target

Only apply the margin to parts of the path that are in the "target" budget mode. The budget mode of the path is determined by the **-adjust_latency** option of the **set_budget_options** command. The "target" budget mode is one of the possible postcts modes. Some blocks may be in the prects mode while others are in postcts modes. By default, the margin is applied in all modes.

-actual

Only apply the margin to parts of the path that are in the "actual" budget mode. The budget mode of the path is determined by the **-adjust_latency** option of the **set_budget_options** command. The "actual" budget mode is one of the possible postcts modes. Some blocks may be in the prects mode while others are in postcts modes. By default, the margin is applied in all modes.

-clock *budget_clock_spec*

Apply the margin to budgeted paths that start from or end at the clocks or part of clocks given in the *budget_clock_spec*. See the "Budget clock specs" section below for details on how to select specific clocks.

-from_clock *budget_clock_spec*

Apply the margin to budgeted paths that start from the clocks or part of clocks given in the *budget_clock_spec*. See the "Budget clock specs" below section for details on how to select specific clocks. This option must be used along with **-to_clock**.

-to_clock *budget_clock_spec*

Apply the margin to budgeted paths that end at the clocks or part of clocks given in the *budget_clock_spec*. See the "Budget clock specs" below section for details on how to select specific clocks. This option must be used along with **-from_clock**.

-corner *corner_name*

Specifies that values given for the other options should be applied in the process corner with the specified name.

-default

Specifies that values given for the other options should be applied in process corners do not have specific values given with the **-corner** option.

DESCRIPTION

Margins specified by this command are added to the clock uncertainty of budgeted paths in the SDC output of the **write_budgets** command. Use **-launch** to specify the margins to be applied at the start of the path. Use **-capture** to specify the margins to be applied at the end of the path. If both are specified, the effective margin for the whole path will be the sum of the two margins.

Various options can be used to limit the circumstances where the margins are applied. Use **-setup** or **-hold** to specify that the margin

should be applied on setup or hold paths.

The **-prects**, **-postcts**, **-target**, or **-actual** options work in conjuction with the **-adjust_latency** option of the **set_budget_options** command. For example, if you use the **-prects** option in **set_budget_margins**, then the given margin will be applied to any block that has "-adjust_latency prects" set by the **set_budget_options** command. If you use the **-postcts** command, the given margin will be applied to any block that has "-adjust_latency target" or "-adjust_latency actual" set by the **set_budget_options** command.

In the case of conflicts between margin constraints, more specific constraints will override more general constraints.

Budget margins are also used by the **compute_budget_constraints** command during automatic budget optimization. Please see the man page for the app option **plan.budget.time_with_margins** for details on how to control this feature.

You can use the **report_budgets -html_dir** command to dump out statistics about your budgets. Margins from this command will be reflected in the "path summaries" in the html report.

This command returns 1 on success, 0 otherwise.

Budget clock specs

When specifying clocks for various budgeting options, you can provide extra information to limit the parameter to a specific subtree of the clock or edge of a clock. For example, you can set a parameter for the subtree of CLK1 which is inside of BLOCK1. And you can set a different parameter for the subtree of CLK1 which is inside of BLOCK2. In general, every time a clock tree passes into or out of a block, this will define a new part to the tree that you can control.

The syntax for specifying a budget_clock_spec is as follows. To specify all parts of a clock (in all blocks), use the clock name:

CLK1

To specify all clocks in all blocks, use "*":

*****

To specify the part of a clock in a selected block, append ":" and the name of the block instance:

CLK1:my_block

To specify all clocks in a selected block, use "*" followed by ":" and the name of the block instance:

***:my_block**

To specify the part of a clock that crosses into or out of a block at a particular block pin, use the clock name, followed by ":" and the name of the block pin. For example, use the following to specify the top-level portion of CLK1 that originates at the clock output of my_block:

CLK1:my_block/CLKOUT

To specify a particular edge of a clock, use a second ":" followed by the word "rise" or "fall". This can be used in combination with any of the specifications above that name a particular clock. For example:

CLK1::rise
CLK1:my_block:fall
CLK1:my_block/CLKOUT:rise

Sometimes it is convenient to specify constraints for a master clock and all of the generated clocks downstream from the master clock. This is refereed to this as a constraint "clock group". To specify a clock group, follow the name of the master clock by a plus (+) character. The specify clock must a be regular clock, not a generated clock. The downstream generated clocks will be implicitly included in the group. Here are the legal usages of clock groups:

CLK1+
CLK1+::rise
CLK1+::fall

If this command is called multiple times for the same clock, the more specific parameter is applied. For example, if you set a parameter for "CLK:my_block" and you set a parameter for "CLK", the parameter for "CLK:my_block" will be used inside my_block and the parameter for "CLK" will be used everywhere else. More general parameters will not overwrite more specific ones.

EXAMPLES

Apply a margin on paths launched from the rising edge clock CLK1 in block B1. Apply only for setup paths and only if the launch block has "-adjust_latency prects" set in the set_budget_options command.

```
prompt> set_budget_margin -prects -setup -launch 0.2 -clock CLK:B1:rise
```

Apply margins on all hold paths between block B2 and block B3. Only apply if the block has "-adjust_latency target" or "-adjust_latency actual" set in the set_budget_options command. A margin of 0.2 will be set in the launch block (B2) and a margin of 0.1 will be set in the capture block (B3).

```
prompt> set_budget_margin -postcts -hold -launch 0.2 -capture 0.1 \
        -from_clock *:B2 -to_clock *:B3
```

SEE ALSO

compute_budget_constraints(2)
report_budget(2)
write_budgets(2)
plan.budget.time_with_margins(3)

set_budget_options

Specifies general options to be used during creation of timing budgets.

SYNTAX

int **set_budget_options**
[-reset]
[-add_blocks]
[-remove_blocks]
[-adjust_latency prects | target | actual]
[-launch_hold_fix true | false]
[-capture_hold_fix true | false]
[-feed_hold_fix true | false]
[-launch_fixed_delay true | false]
[-capture_fixed_delay true | false]
[-feed_fixed_delay true | false]
[-min_segment_percent percent]

[-top_level]
[-all]
[*instances*]

Data Types

instances collection
percent float

ARGUMENTS

-reset

Removes all previously specified budget constraints from the current design. This includes constraints and options from **set_budget_options**, **set_pin_budget_constraints**, **set_latency_budget_constraints**, **set_boundary_budget_constraints**, and **compute_budget_constraints**.

-add_blocks

Specifies that budgeting commands should create a budget for the specified block instances. This option must be accompanied by a list of instance path names to add.

-remove_blocks

Specifies that blocks previous added by the *-add_blocks* option should no longer be considered for budgeting. This option must be accompanied by a list of instance path names to remove or the *-all* option.

-adjust_latency prects | target | actual

This option controls how clock alignment is handled at the boundary of budgeted blocks. At the time your block budgets are written (in the **write_budgets** command), it is possible that the clocks of your designs are not yet balanced. For example, you may have run clock tree synthesis in one of your blocks and not another. Or perhaps, you have run clock tree synthesis in each of your blocks, but have not yet balanced the blocks with each other.

By setting *-adjust_latency* for your blocks, you can create block budgets that offset the current clock imbalances between the blocks and the top level. Constraints will be created that assume all of your clocks will eventually be optimized to have budget values similar those set manually by the *-early_latency* and *-late_latency* options of the **set_latency_budget_constraints** command -- or set automatically by the **compute_budget_constraints** command.

It is important to choose the correct adjust_option for *-adjust_latency*. The correct choice will depend on the state of your design in the design flow. The choices are outlined below. If -adjust_latency has never been specified for a block, the "prects" style of adjustment will be used, which is most appropriate for early in the design flow. It is OK to use a "prects" budget throughout the entire design flow, but this early budget will not be adapted to changes in your clocks that occur during CTS.

You should specify the blocks to apply this option to by specifying an instance list or using the *-all* option.

Refer to the **write_budgets** man page for more examples of the recommended use of the *-adjust_latency* option.

prects

When your block's I/O constraints are generated, the latency of the virtual clocks that define the block's I/O timing will be adjusted to match the current clock latencies in your block. The internal clock latency of the block will not be adjusted. For example, if the block currently has small clock latency and the budget latency target is large, the virtual clocks at the block's boundary will be adjusted to have short latencies that are similar to those inside of the block. You can use the *-latency* option of the **report_budget** to see the adjustment being applied. You should use "prects" latency adjustment when your block is using ideal (not propagated) clocks.

target

The internal latency of your block will be adjusted to match the target top-level budget constraint, as set by **set_latency_budget_constraints** or **compute_budget_constraints**. This adjustment will be accomplished by resetting the source latency on the clocks entering your block. For example, if the block currently has small clock latency and the budget latency target is large, the source latency on the clocks entering your block will be increased. You can use the *-latency* option of the **report_budget** to see the adjustment being applied. You should use "target" latency adjustment when your block has propagated clocks.

actual

The internal latency of your block will be adjusted to match the actual propagated top-level latency. This adjustment will be accomplished by propagating latency on your top-level clock until it reaches the block boundary. The resulting latency will be directly applied as the source latency for your block clock. Since the budgeted top-level latencies are ignored during this adjustment, you should only use "actual" adjustment after your top-level clocks have been balanced to the desired values.

Note that your datapath budgets will still use budget latency targets to determine the budgeted path length. You should use **compute_budget_constraints -latency_targets actual balance_latency false** to readjust your budget latency targets before you call **write_budgets**.

-launch_hold_fix true | false

This option affects the operation of the **write_budgets** command when outputting hold constraints. When set to false, the **write_budgets** command will create constraints that avoid the insertion of hold buffers on the output of selected blocks. You should specify the blocks to apply this option to by specifying an instance list, *-top_level*, or the *-all* option. The default value is true.

-capture_hold_fix true | false

This option affects the operation of the **write_budgets** command when outputting hold constraints. When set to false, the

write_budgets command will create constraints that avoid the insertion of hold buffers on the input of selected blocks. You should specify the blocks to apply this option to by specifying an instance list, *-top_level*, or the *-all* option. The default value is true.

-feed_hold_fix true | false

This option affects the operation of the **write_budgets** command when outputting hold constraints. When set to false, the **write_budgets** command will create constraints that avoid the insertion of hold buffers on the part of paths that feed through selected blocks. You should specify the blocks to apply this option to by specifying an instance list, *-top_level*, or the *-all* option. The default value is false.

-launch_fixed_delay true | false

This option affects the operation of the **compute_budget_constraints** command. By default, the budgeting algorithm assigns slack to all blocks along a top level path. If the top-level path has negative slack, then each block is assigned part of that negative slack to fix. By setting this option to true, you are telling the budgeter that the output of the selected block is "fixed". The budgeter will treat this part of the path as it would treat a hard macro. The budgeter will not assign negative slack to this part of the path, because it will assume that the block will not be optimized. Likewise, the budgeter will assign only a very small fraction of any positive slack to this part of the path. You should specify the blocks to apply this option to by specifying an instance list, *-top_level*, or the *-all* option. The default value is false.

-capture_fixed_delay true | false

This option affects the operation of the **compute_budget_constraints** command. By default, the budgeting algorithm assigns slack to all blocks along a top level path. If the top-level path has negative slack, then each block is assigned part of that negative slack to fix. By setting this option to true, you are telling the budgeter that the input of the selected block is "fixed". The budgeter will treat this part of the path as it would treat a hard macro. The budgeter will not assign negative slack to this part of the path, because it will assume that the block will not be optimized. Likewise, the budgeter will assign only a very small fraction of any positive slack to this part of the path. You should specify the blocks to apply this option to by specifying an instance list, *-top_level*, or the *-all* option. The default value is false.

-feed_fixed_delay true | false

This option affects the operation of the **compute_budget_constraints** command. By default, the budgeting algorithm assigns slack to all blocks along a top level path. If the top-level path has negative slack, then each block is assigned part of that negative slack to fix. By setting this option to true, you are telling the budgeter that parts of the path that feed through the selected block are "fixed". The budgeter will treat this part of the path as it would treat a hard macro. The budgeter will not assign negative slack to this part of the path, because it will assume that the block will not be optimized. Likewise, the budgeter will assign only a very small fraction of any positive slack to this part of the path. You should specify the blocks to apply this option to by specifying an instance list, *-top_level*, or the *-all* option. The default value is false.

-min_segment_percent percent

This option affects the operation of the **compute_budget_constraints** command. When it is used, the budgeting computation tries to assign at least the indicated percentage of the clock cycle to all the budget segments in a path if possible. The percent number given should be less or equal to 0.5. If the percent number 0 is given, it has the effect of removing any previously set *min_segment_percent* constraints. This is a low priority constraint that will be taken into account only if higher priority constraints can also be satisfied. Higher priority constraints include zero-slack budgeting, fixed delay budgets and user-specified budgets. This option cannot be used together with the options **-top_level** or with a list of instances.

-top_level

Specify that the given option should be applied to the top level of your design, not the blocks.

-all

Use with other options of this command to operate on all blocks and the top_level of your design. The blocks that are used are the same ones that were previously added with the *-add_blocks* option.

instances

Specify a list of instance names for blocks of your design. Use with other options of this command to operate only on specific blocks of your design. The instances specified must be added previously by using the *-add_blocks* option.

DESCRIPTION

Specifies general options to be used during creation of timing budgets. To get a summary of the options you have set, use "**report_budget -blocks**". To set specific budget parameters, use the **set_pin_budget_constraints**, **set_segment_budget_constraints**, **set_latency_budget_constraints**, and **set_boundary_budget_constraints** commands.

Use the **write_budgets** command to generate SDC that can be applied to lower level blocks. The **compute_budget_constraints** can be used to set key budget parameters automatically. The **report_budget** command will generate useful information about the current budget calculation, allocation, and whether that budget is currently being met. Use "**write_script -include budget**" to save a Tcl script that includes your current settings from this command.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following example declares three blocks in the current design which will be budgeted:

```
prompt> set_budget_options -add_blocks { core/CPU core/MMU exif }
```

The following example removes one block from the list of blocks to be budgeted:

```
prompt> set_budget_options -remove_blocks exif
```

The following example marks core/CPU to have fixed timing. The **compute_budget_constraints** command will treat this block as if it were a hard macro.

```
prompt> set_budget_options -launch_fixed_delay true \
  -capture_fixed_delay true -feed_fixed_delay true core/CPU
```

The following example marks core/CPU as having complete clock trees. When the budget is written by the **write_budgets** command, clock latencies at the block boundary will be adjusted accordingly. Also, the skew margin for the block will no longer be reserved.

```
prompt> set_budget_options -adjust_latency target core/CPU
```

Refer to the **write_budgets** man page for more examples of using the *-adjust_latency* option.

SEE ALSO

compute_budget_constraints(2)
report_budget(2)
set_boundary_budget_constraints(2)
set_budget_margins(2)
set_latency_budget_constraints(2)
set_pin_budget_constraints(2)

write_budgets(2)
write_script(2)

set_budget_shell_latencies

Creates special latencies for clocks that are generated in a budget shell or pass through a budget shell.

SYNTAX

int **set_budget_shell_latencies**
-block *block_module_name*
[-clock *clock_name*]
[-block_mode *block_mode_name*]
[-block_output *block_port_name*]
[-corner *corner_name*]
[-rise]
[-fall]
[-early]
[-late]
delay_value

Data Types

block_module_name string
clock_name string
block_mode_name string
block_port_name string
corner_name string
delay_value float

ARGUMENTS

-block *block_module_name*

Specifies the name of the budget shell block. This is the name of the module, not the instance of the module.

-clock *clock_name*

Applies the latency only for the specified top-level clock. By default, the latency is applied to all clocks.

-block_mode *block_mode_name*

Applies the latency only for clocks that are associated with the given operating mode of the block. By default, the latency is applied to clocks for all modes.

-block_output *block_port_name*

Applies the latency only for clocks leaving the block at the given output port. By default, the latency is applied to clocks leaving the block at any port.

-corner *corner_name*

Applies the latency only for the given process corner. By default, all corners are used.

-rise

Applies the latency only for the rising edge of the clock at the block output. By default, both rise and fall are used.

-fall

Applies the latency only for the falling edge of the clock at the block output. By default, both rise and fall are used.

-early

Applies the delay only for the best case latency. By default, both early and late are used.

-late

Applies the delay only for the worst case latency. By default, both early and late are used.

delay_value

Specifies the value set for the latency.

DESCRIPTION

This command creates budget shell latencies. Budget shells are created by the **-shell_subblocks** option of the **write_budgets** command. If the subblock in question has internal clocks of one of the following types, it is necessary to specify the latency of that clock.

- "Feedthrough clock": If a budget shell block has a top-level clock feeding through it, the **set_budget_shell_latencies** command specifies the part of the clock latency that is inside of the block.
- "Clock source in the block": If a budget shell block has a clock whose source is in the block, the **set_budget_shell_latencies** command specifies the part of the clock latency that is inside of the block.
- "Generated clock in the block": If a budget shell block has a generated clock source in the block and the master clock is outside of the block, the **set_budget_shell_latencies** command specifies the latency between the point where the master clock enters the block and the generated clock leaves the block.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following example sets the shell latency for any clocks that leave the block at port "clock_out".

```
prompt> set_budget_shell_latencies -block BLOCK -block_output clock_out
2.0
```

SEE ALSO

compute_budget_constraints(2)
report_budget(2)
set_boundary_budget_constraints(2)
set_budget_options(2)
set_latency_budget_constraints(2)
set_pin_budget_constraints(2)
write_budgets(2)
write_script(2)

set_bundle_pin_constraints

Sets pin constraints on net bundles defined with the **create_bundle** and **create_bundles_from_patterns** commands.

SYNTAX

status **set_bundle_pin_constraints**
[-bundles *bundles*]
[-cells *cells*]
[-self]
[-keep_pins_together true | false]
[-range {*start end*}]
[-bundle_order ordered | increasing | decreasing | equal-distance]
[-allow_feedthroughs true | false]
[-allowed_layers *layers*]
[-pin_spacing *spacing*]
[-pin_spacing_distance *distance*]
[-sides *side_numbers*]
[-width *width*]
[-length *length*]

Data Types

bundles collection of net bundles
cells collection cells
start integer
end integer
layers collection of layers
spacing float
distance float
side_numbers collection of integers
width real or list of layer-real pairs
length real or list of layer-real pairs

ARGUMENTS

-bundles *bundles*

Limits the constraints to the specified net bundles. The bundles must be in the top-level of the current design. If not specified, the command applies the constraints to all net bundles.

-cells *cell_collection*

Limits the constraints to the specified cells. For example, you can combine the **-cells** and **-allowed_layers** options to restrict the pins to only specified layers for the specified cell. You can combine the **-cells** and **-self** options. The cells must be in the top-level

of the current design. If both the **-cells** and **-self** options are unspecified, the constraints apply to all block cells.

-self

Applies the constraints to the top-level block. You can combine the **-cells** and **-self** options. If both the **-cells** and **-self** options are unspecified, the constraints apply to all block cells.

-keep_pins_together true | false

Specify whether bundle pins are kept together. When this option is true, all bundle pins on the same block are placed on the same layer (hard constraint) and no other pins should be placed between the bundle pins (soft constraint). In certain cases, the command might violate the soft constraint and insert other pins between the bundle pins. The default value is false.

-bundle_order ordered | increasing | decreasing | equal-distance

Specifies the relative placement order of the pins connected to the bundle nets. Note that the order refers to the net order in the bundle, not the pins on the block. The bundle order is measured as seen at the current top design. Pin placement considers this option only when option **-keep_pins_together** is set to true. If two or more bundles connecting the same collection of MIB pins but with conflict orders, only one of them can be honored.

- ordered: Pins are ordered according to the net order in the bundle. For vertical block sides, bundle pins can be placed either from bottom to top or from top to bottom. For horizontal sides, bundle pins can be placed either from left to right or from right to left. The orders on each edges are chosen such that for the bundle nets connect the bundle pins from one edge to another edge, there is no net crossing, i.e. river turn style.

  In the following example, bundle pin constraint is specified as *ordered* so that the nets connecting two bundle pins are of the river turn style, i.e. no crossing.

  ```
  prompt> set_bundle_pin_constraints -allowed_layers {M3}\
    -bundles bundle1 -bundle_order ordered -keep_pins_together true
  ```

- increasing: Pins are ordered according to the increasing net order in the bundle. For vertical block sides, bundle pins are placed from bottom to top. For horizontal block sides, bundle pins are placed from left to right.

  In the following example, bundle pin constraint is specified on a rectangular block so that bundle pins are placed on side 2 (horizontal edge) of block. Pins are placed from left to right (Least significant bit is the leftmost; Most significant bit is the right most on side2).

  ```
  prompt> set_bundle_pin_constraints -bundles bundle1 -sides 2 \
    -bundle_order increasing -keep_pins_together true
  ```

- decreasing: Pins are ordered according to the decreasing net order in the bundle. For vertical block sides, bundle pins are placed from top to bottom. For horizontal block sides, bundle pins are placed from right to left.

  In the following example, bundle pin constraint is specified on a rectangular block so that bundle pins are placed on side 2 (horizontal edge) of block. Pins are placed from right to left (Most significant bit is the leftmost; Least significant bit is the rightmost).

  ```
  prompt> set_bundle_pin_constraints -bundles bundle1 -sides 2 \
    -bundle_order decreasing -keep_pins_together true
  ```

- equal-distance: Pins are ordered as with **-bundle_order ordered**, but the pin ordering for two edges is chosen to create the same Manhattan distance between each corresponding pair of pins.

  In the following example, bundle pin constraint is specified as *equal-distance* so that the Manhattan distacnes of each bits of the bundle are the same for bundle *bundle1*.

  ```
  prompt> set_bundle_pin_constraints -allowed_layers {M3}\
    -bundles bundle1 -bundle_order equal-distance -keep_pins_together true
  ```

  Note that when setting bundle pin constraints on pins of multiple instantiated blocks (MIB), as the orientation of MIBs can be

different, the tool chooses one of the MIBs randomly to honor the constraints. Therefore the bundle constraints could be not honored for the other MIBs.

When option **-keep_pins_together** is set to true but option **-bundle_order** is not specified, the tool will choose from either **increasing** or **decreasing** based on alignment, abutment and wire length cost. The bundle order might also be decided by already placed or fixed bundle pins on this bundle due to alignment or abutment.

-allow_feedthroughs true | **false**

Specifies whether feedthrough ports can be created in pin placement flow. When feedthroughs are allowed, the global router creates feedthrough routing on a block cell as needed for the selected nets. Each top-level net for which a feedthrough port is created is split into a set of new top-level nets and child-level nets, if feedthrough nets are created for the child nets. Directions are assigned for the newly created feedthrough ports. Because new ports are created in the block cell, the netlist is modified as well.

By default, the feedthrough is not allowed for bundle nets.

-allowed_layers ***metal_layers***

Specifies the set of metal layers allowed for pin placement. The argument is a list of metal layer names as returned by the **get_layers** command.

By default, the metal layers from M2 to the maximum metal layer (specified earlier during floorplan flow) are allowed for pin placement. If the maximum metal layer has not been specified, the maximum available metal layer specified in the technology file is used as the default. If the maximum layer specified is beyond the maximum metal layer for the current design, then the maximum layer in the technology file is used.

-pin_spacing ***spacing_number***

Specifies the minimum number of wire tracks between adjacent pins or between a pin and a preroute wire that cuts across a block edge in the normal direction. Pins that are created are always snapped to wire tracks. The default pin-to-pin spacing is one wire track. The spacing number must be 0 or a positive integer.

-pin_spacing_distance ***spacing_distance***

Specifies the minimum distance between adjacent pins or between a pin and a preroute wire that cuts across a block edge in the normal direction. Pins that are created are always snapped to wire tracks.

-sides ***side_numbers***

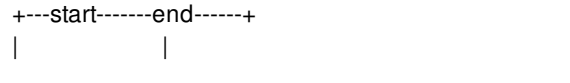
Specifies one or more block sides on which the pin must be placed.

The side numbers are a list of positive integers that start from 1. The left edge of a rectangular shape, or the lower left-most vertical edge of a rectilinear shape, is side number 1. The side numbers increment as you proceed clockwise around the shape. You cannot specify a value of 0 for this option.

-range {*start end*}

Specifies the range of positions on a side within which the bundle pins must be placed. The positive *start* and *end* values refer to the distances as measured to the starting point of the edge in microns (or the default unit). The negative *start* and *end* values refer to the distances as measured to the ending point of the edge in microns (or the default unit). Postive *start* is the position that is closest to the edge's starting point, and postive *end* is the position that is farthest to the edge's starting point. Vice versa for negative *start* and *end* positions. The starting point of an edge is the vertex of the edge where the edge begins as you proceed clockwise around the block's boundary from the lowest vertex of the leftmost edge. If there are more than one leftmost edges, then from the lowest one.

The following simple drawing shows an example of how postive *start* and *end* are measured on a rectangular block on each of the sides.



```
 |                start
end                  |
 |                 |
 |                end
start                |
 |                 |
 +------end-----start-----+
```

The following simple drawing shows an example of how negative *start* and *end* are measured on a rectangular block on each of the sides.

```
 +---end-------start------+
 |                 |
 |                end
start                |
 |                 |
 |                start
end                  |
 |                 |
 +------start-----end-----+
```

-width *pin_width*

Specifies the width of the pin. The width is perpendicular to the layer's preferred routing direction. The syntax can be a single real number to indicate the width is applied to all allowed layers (referred to as common pin width). Or alternatively, the width can be specified as a list of layer-real pairs as the following (referred to as per layer pin width):

-width {{M2 0.2} {M3 0.2} {M4 0.3} {M5 0.3}}

It means that if the pin is to be placed on layer M2 or M3, its width should be 0.2 and if on M4 or M5, 0.3. On other layers, use the default width defined by the tech file.

If you first set the pin width to be common pin width, then set the pin width to be per layer width with a second **set_bundle_pin_constraints** command, then the width specified in the second command will override the first. Likewise, if you first set the pin width to be per-layer width, then set the pin width to be a common pin width with a second **set_bundle_pin_constraints** command, then the width specified in the second command will override the first.

However, if you first set the pin width to be per layer, then in a second command sets the pin width again to be per layer but with different layers or different values, then the combined per layer pin width will be the final constraints. For example, consider the following example:

```
set_bundle_pin_constraints -bundles {bundle1} -width 0.3
set_bundle_pin_constraints -bundles {bundle1} -width {{M2 0.3} {M3 0.3}}
```

The first common pin width constraint will be overridden by the second per layer pin width. The pin width should be 0.3 on layer M2 or M3, but default width on all other layers.

And for the following example:

```
set_bundle_pin_constraints -bundles {bundle1} -width {{M3 0.4} {M4 0.4}}
set_bundle_pin_constraints -bundles {bundle1} -width {{M2 0.3} {M3 0.3}}
```

The pin width should be 0.3 on layer M2 or M3, 0.4 on layer M4, and default width on all other layers.

-length *pin_length*

Specifies the length of the pin. The length is in parallel to the layer's preferred routing direction. The syntax can be a single real number to indicate the length is applied to all allowed layers (referred to as common pin length). Or alternatively, the length can be specified as a list of layer-real pairs as the following (referred to as per layer pin length):

-length {{M2 0.2} {M3 0.2} {M4 0.3} {M5 0.3}}

It means that if the pin is to be placed on layer M2 or M3, its length should be 0.2 and if on M4 or M5, 0.3. On unspecified layers, the pin length is derived from the pin width. Refer to the manpage of **place_pins** for details.

DESCRIPTION

This command constrains the placement of net bundles on a physical design block. The constraints are honored by the **place_pins** command. The constraints specify the block side, pin ordering, and other pin placement details.

The **set_bundle_pin_constraints** command is additive. The constraint values set in previous calls are retained, unless they are explicitly overridden by subsequent calls.

Constraints can apply to certain cell instances only. The cell specified with the *-cells* option must be a physical block and must not be a library cell or logical block. The constraint can apply to the top-level block of the current design by specifying the **-self** option.

The constraints can apply to either a specific bundle or to all bundles. Also, the constraints can apply to a specific block cell or to all block cells. There are four possible combinations of bundle and block constraints, ordered from general to specific.

The **set_editability** command can enable or disable the **set_bundle_pin_constraints** command for specified blocks or levels of physical hierarchy. This command has no effect on the blocks that are disabled with the **set_editability** command.

- Global constraints: Constraints apply to all bundles and all block instances.
- Specific bundle constraints: Constraints apply to a specific bundle and all block instances.
- Specific block constraints: Constraints apply to all bundles and a specific block instance. Note that this can apply to the top-level block when you specify the *-self* option.
- Bundle and block pair constraints: Constraints apply to a specific bundle and a specific block instance. Note that this can apply to the top-level block when you specify the *-self* option.

Note that bundle and block pair constraints override the other three types; block constraints override bundle and global constraints, and so on.

Unless otherwise mentioned, all types of constraints can be applied in each of these situations.

The command can only be used to set constraints on bundles and cells in the top-level design, not within the blocks. To set constraints within a block, use the **current_design** command to make the block the current design and apply the constraint.

The **set_editability** command can enable or disable the **set_bundle_pin_constraints** command for specified blocks or levels of physical hierarchy. This command has no effect on the blocks that are disabled with the **set_editability** command.

This command returns 1 on success, 0 otherwise.

Please note, starting from 2019.12 release, a new command **create_pin_constraint** is introduced that can also creates the bundle pin constraints (as well as individual pin constraints) and has similar command line options. The major difference is that **create_pin_constraint** returns a collection of constraints it created, which under certain scenarios might be desired.

EXAMPLES

The following example sets the allowed pin layers to METAL2 and METAL3 for bundle BUNDLE_0. This creates a specific bundle constraint.

```
prompt> set_bundle_pin_constraints \
  -allowed_layers [get_layers METAL2 METAL3] \
  -bundles [get_bundles BUNDLE_0]
```

The following example enforces ordering on all bundle pins of cell BLOCK1. This creates a specific block constraint.

```
prompt> set_bundle_pin_constraints -bundle_order ordered \
  -cells [get_cells BLOCK0] -keep_pins_together true
```

The following example enforces ordering and minimum pin spacing for bundle BUNDLE_0. This creates a specific bundle constraint.

```
prompt> set_bundle_pin_constraints -bundle_order ordered \
  -pin_spacing 2 -bundles [get_bundles BUNDLE_0] -keep_pins_together true
```

SEE ALSO

place_pins(2)
report_bundle_pin_constraints(2)
remove_bundle_pin_constraints(2)
set_individual_pin_constraints(2)
set_block_pin_constraints(2)
set_editability(2)
create_pin_constraint(2)
remove_pin_constraints(2)

set_busplan_constraints

Sets constraints on pipeline register counts for buses during bus planning.

SYNTAX

int **set_busplan_constraints**
-from *buses*
[-to *buses*]
[-to_value *constraint_value*]
[-type *constraint_type*]
[-remove]

Data Types

buses	collection of buses
constraint_value	integer
constraint_type	string

ARGUMENTS

-from *buses*

Specifies the buses on which to set or remove constraints.

-to *buses*

Specifies the buses on which the constraint is to created to. This option is mutually exclusive with *-to_value* and *-remove*.

-to_value *constraint_value*

Specifies the value of the constraint for constraints that accept a specific value. This option is mutually exclusive with *-to* and *-remove*.

-type *constraint_type*

Specifies the type of constraint to apply. There are 5 valid constraints: *less*, *less_or_equal*, *greater*, *greater_or_equal*, and *equal*. These constraints can also be represented by the following symbols: <, <=, >, >=, and =. These constraints can be applied between 2 buses using *-from* and *-to* or can be applied on a single bus using *-from* and *-to_value*.

-remove

Removes any constraints on the *-from* bus. This option is mutually exclusive with *-to* and *-to_value*.

DESCRIPTION

This command defines constraints between buses by restricting the number of registers allowed on each bus. There are 5 valid constraints: *less*, *less_or_equal*, *greater*, *greater_or_equal*, and *equal*. These constraints can also be represented by the following symbols: <, <=, >, >=, =.

Constraints can be applied between 2 buses using *-from* and *-to* or can be applied on a single bus using *-from* and *-to_value*. Constraints can be removed using *-from* and *-remove*.

EXAMPLES

The following example creates a constraint that requires bus1 and bus2 have the same number of pipeline registers. Note that this example assumes bus1 and bus2 are created previously with the *create_busplans* command.

```
prompt> set_busplan_constraints -from bus1 -to bus2 -type =
1
```

The following example creates a constraint that requires bus1 have less than 5 pipeline registers.

```
prompt> set_busplan_constraints -from bus1 -to_value 5 -type <
1
```

SEE ALSO

create_busplans(2)
get_busplans(2)
modify_busplan(2)
report_busplan_constraints(2)
report_busplans(2)
write_busplans(2)

set_case_analysis

Specifies that a port or pin is at a constant logic value 1 or 0, or is considered with a rising or falling transition.

SYNTAX

string **set_case_analysis** *value*
port_or_pin_list

string *0* | *1* | *rising* | *falling*
list *port_or_pin_list*

ARGUMENTS

value

Specifies a constant logic value or a transition to assign to the given pin or port. The valid constant values are **0**, **1**, **zero**, and **one**. The valid transition values are **rising**, **falling**, **rise**, and **fall**.

port_or_pin_list

Lists ports or pins to which the case analysis is assigned. In the case of pins, constant propagation is executed forward only. No backward constant propagation is performed.

DESCRIPTION

Case analysis is a way of specifying a given mode for the design without altering the netlist structure. You can specify for the current timing analysis session, that some signals are at a constant value (**1** or **0**), or that only one type of transition (**rising** or **falling**) is considered.

Pins of a design may be driven by logic values from the design, but if case analysis is used to set a value on such pins, the values set by case analysis will dominate. If the case values are subsequently removed, the value driving the pin from the design will be propagated.

When case analysis is specified as a constant value, this value is propagated through the network as long as the constant value is a controlling value for the traversed logic. For example, if you specify that one of the inputs of a NAND gate is a constant value **0**, it is propagated to the NAND output, which is now considered at a logic constant **1**. This propagated constant value is then propagated to all cells driven by this signal.

In the event that the case analysis value is a transition, the given pin or port is considered only for timing analysis with the specified transition. The other transition is disabled. The case analysis information is used by all analysis commands.

Case analysis is used in addition to the mode commands to fully specify the mode of a design. For example, a design that

instantiates models with a TESTMODE, is specified so that the TESTMODE is disabled during the timing analysis session by using the **set_mode** command. In addition, if a TESTMODE signal exists on the design, it is specified to a constant logic value, so that all test logic controlled by the TESTMODE signal is disabled.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example shows that the port *IN1* is at a constant logic value of **0**.

```
prompt> set_case_analysis 0 IN1
```

The following example shows that the pins *U1/U2/A* and *U1/U3/CI* are considered only for a *rising* transition. The *falling* transition on these pins is disabled.

```
prompt> set_case_analysis rising {U1/U2/A U1/U3/CI}
```

The following example specifies how to disable the TESTMODE of a design for which instances are models having a TESTMODE mode. The design also has a *TEST_PORT* port set to a constant logic value **0**.

```
prompt> remove_mode TESTMODE U1/U2
prompt> set_case_analysis 0 TEST_PORT
```

SEE ALSO

remove_case_analysis(2)
report_case_analysis(2)
disable_case_analysis(3)
set_mode(2)

set_cell_hierarchy_type

Change the hierarchy type of a hierarchical instance and optionally add a boundary, if specified.

SYNTAX

collection **set_cell_hierarchy_type**
-type *hierarchy_type*
[-boundary { { {*llx lly*} {*urx ury*} } |
{ {*x y*} {*x y*} {*x y*} {*x y*} ... } } |
geometric_objects]
cell

Data Types

hierarchy_type string
llx float
lly float
urx float
ury float
x float
y float
geometric_objects collection
cell A single hierarchical cell

ARGUMENTS

-type *hierarchy_type*

New hierarchy_type for the cell. Valid values are normal or boundary. This option is required.

-boundary { {{*llx lly*} {*urx ury*}} | {{*x y*} {*x y*} {*x y*} {*x y*}...} } | geometric_objects

Specifies the boundary of the cell. The boundary may be a rectangle or a polygon. A rectangle is specified by its lower-left and upper-right coordinates (i.e., {*llx lly*} {*urx ury*}). A polygon is specified by its points (i.e., {*x y*} {*x y*} {*x y*} {*x y*}...).

Polygons may also be specified as the combined area of a heterogenous collection of objects with physical geometry, such as poly_rects, geo_masks, shapes, layers, and other physical objects. In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

When specifying the boundary polygon as a collection of physical objects, the resulting area must resolve to a single, connected polygon. It is an error to specify a collection of objects with a combined area that resolves to multiple polygons.

This is an optional option.

cell

Specifies the hierarchical cell whose hierarchy_type is to be changed. It can be specified as the cell name or a collection containing one cell object.

This option is required.

DESCRIPTION

This command takes the name of a hierarchical (not leaf) cell or a collection containing a single cell and changes the hierarchy_type to the specified type. If the optional boundary argument is given it is used to set the boundary, if the type is changing to boundary or to create the initial boundary for the new hierBoundary object, if the type is boundary.

The command will fail if the given path doesnt exist, or is a lib-cell (non-hierarchical cell).

EXAMPLES

The following example creates a hierBoundary out of the cell found at path a/b/c with a rectangular boundary from (0, 0) through (1000, 1000).

prompt> **set_cell_hierarchy_type -type boundary -boundary {{0 0} {1000 1000}} a/b/c**

The following command will turn the cell a/b/c, into a logical hierarchical cell

prompt> set_cell_hierarchy_type -type normal a/b/c

SEE ALSO

get_cells(2)
commit_block(2)
uncommit_block(2)

set_cell_location

Specifies the physical location for leaf cells.

SYNTAX

status **set_cell_location**
[-design *design*]
-coordinates *location*
[-z_offset *value*
[-ignore_fixed]
[-orientation *orient*]
[-fixed]
cell_list

Data Types

location	point
value	int
orient	string
cell_list	collection

ARGUMENTS

-coordinates *point*

Specifies the lower-left coordinates of the boundary of the specified cells. The numbers are in microns relative to the chip origin.

-z_offset *value*

Specifies the stack_order/z-offset of the physical cell in a 3DIC design

-ignore_fixed

Sets the location for cells even if they have fixed placement status.

-fixed

Sets the **physical_status** attribute on the specified cells to fixed to prevent the tool from moving these cells.

-orientation *orient*

Specifies the new orientation.

Valid values are

- **R0**: Nominal orientation (north)

- **R90**: 90-degree counterclockwise rotation (west)
- **R180**: 180-degree rotation (south)
- **R270**: 270-degree counterclockwise rotation (east)
- **MY**: reflection in the y-axis (flipped north)
- **MX**: 180-degree rotation followed by reflection in the y-axis (flipped south)
- **MYR90**: 90-degree counterclockwise rotation followed by reflection in the y-axis (flipped east)
- **MXR90**: 270-degree counterclockwise rotation followed by reflection in the y-axis (flipped west)

cell_list

Specifies the affected leaf cells. Typically, this command uses a single leaf cell as the argument.

DESCRIPTION

The **set_cell_location** command sets the physical location for the specified leaf cells. The specified location is used as the lower-left corner of the boundary of the cell.

This command can also set the orientation for the specified cells and change their **physical_status** attribute to **fixed**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the location of the lower-left corner of the cell named INST_1 to (100 100):

```
prompt> set_cell_location -coordinates { 100 100 } INST_1
```

SEE ALSO

set_attribute(2)
move_objects(2)

set_cell_mode

Specifies the active cell mode for cell instances

SYNTAX

Boolean **set_cell_mode**
mode_list
cell_list
list *mode_list*
list *cell_list*
[-quiet]

ARGUMENTS

mode_list

Specifies a list of modes, each of which is to be made the active mode for its mode group.

cell_list

Specifies a list of instances for which the specified cell modes are to be made active. This option is required.

-quiet

Suppresses warning and error messages if objects do not have the mode specified. Syntax error messages are not suppressed.

DESCRIPTION

Selects the active mode for a mode group or for several mode groups and disables modes in the same group as the selected active mode. For moded-ETM the mode group **etm** must have one mode enabled to work correctly. The next timing update will give a warning if that is not the case, and report_cell_mode -missing will display mode-ETM cells that lack any set_cell_mode settings.

The moded-ETM is created by the library manager ETM flow.

It is possible for cell mode groups are defined in the .lib library source. Cell like this have slightly different rules from moded-ETMs where the mode group **etm** is added by the Library Manager. For .lib mode groups if no set_cell_mode is given all the modes are enabled. For moded-ETMs where no set_cell_mode is given all arcs are disabled.

Each library cell that is not an moded-ETM can have a set of cell mode groups. Each of these cell mode groups can have two or more cell modes. Each of these cell modes can be mapped to a set of timing arcs of the library cell. When a cell mode is made active for a given instance of the library cell, the cell mode is enabled and all of its timing arcs are enabled for that cell. All other cell

modes are automatically disabled. The timing arcs of the disabled cell modes are then automatically disabled. In the special case of an arc having multiple cell modes mapped to it, the arc will be enabled if any of the modes are enabled.

Note that the cell "modes" managed by this command have nothing to do with "modes" created on a design with the **create_mode** command.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following command selects the READ cell mode as the active mode for the RAM instance Uram1.

prompt> **set_cell_mode READ Uram1**

SEE ALSO

reset_cell_mode(2)
report_cell_mode(2)

set_cell_site

Sets the specified site definition on the cells that match the site definition's size.

SYNTAX

status **set_cell_site**
-site_def *site_name*
[-library *lib_name*]
[-cells *cell_list*]
[-height_type single | multiple]

Data Types

lib_name string
cell_list list
site_name string

ARGUMENTS

-library *lib_name*

Specifies the reference library that contains the library cells. The specified library must be a complete in-memory reference library. It cannot be a pure logic library, such as a library read by the **read_db** command.

If you do not specify this option, the command uses the current library.

-cells *cell_list*

Specifies the cells for which to set the site definition. The specified cells must exist in the reference library.

If you do not specify this option, the command uses all cells in the reference library.

-site_def *site_name*

Specifies the site definition to set on the matching cells. The specified site definition must be defined in the library's technology data.

-height_type single | multiple

Specifies how to match the cell height with the site definition height.

Valid values are

- **multiple** (the default)

  The cell height can be an integer multiple of the site definition height.

- **single**

  The cell height must be the same as the site definition height.

DESCRIPTION

The **set_cell_site** command sets the specified site definition on the design and frame views of the standard cells that match its size. By default, the command applies to all standard cells in the reference library. To restrict the set of cells, use the **-cells** option. The command ignores cells whose **design_type** attribute is **macro**, **black_box**, **module**, **pad**, **pad_spacer**, **flip_chip_driver**, or **corner_pad**.

A cell matches the site definition's size if

- The cell's width is an integer multiple of the site definition's width.
- The cell's height is either an integer multiple of the site definition's height or the same as the site definition's height, depending on the value of the **-height_type** option.

If the cell size does not match the site definition size, the command issues a warning and does not apply the site definition.

EXAMPLES

The following example sets the site definition to unit for the standard cells in the current library whose height is an integer multiple of the site definition's height.

```
prompt> set_cell_site -site_def unit
1
```

The following example sets the site definition to unit for all standard cells named AO* in the lib1 reference library whose height is the same as the site definition's height.

```
prompt> set_cell_site -site_def unit \
  -library lib1 -cells [get_attribute [get_lib_cells lib1/AO*] name] \
  -height_type single
```

SEE ALSO

current_lib(2)
set_attribute(2)
get_attribute(2)
get_site_defs(2)

set_cell_vt_type

This sets the VT type of a library cell that is used to determine which filler is inserted when the create_vtcell_fillers command is issued.

SYNTAX

status **set_cell_vt_type**
-lib_cells *lib_cells*
-vt_type *vt_type*
[-silent]

Data Types

lib_cells	collection
vt_type	string

ARGUMENTS

-lib_cells *lib_cells*

Specifies the library cells that will be marked with the given VT type.

-vt_type *vt_type*

Specifies the VT type to mark the library cells with. The string default is keyworded to be the default VT type.

-silent

When this option is specified the command prints minimal information about the current VT settings.

DESCRIPTION

This sets the VT type of a library cell that is used to determine which filler is inserted when the create_vtcell_fillers command is issued.

EXAMPLES

The following example shows how specify the invx1 cell to be the default VT type and all cells that end with _vtA to have a VT type of

vtA.

```
prompt> set_cell_vt_type -lib_cells "*/*_vtA"  -vt_type vtA
prompt> set_cell_vt_type -lib_cells "mylib/invx1"  -vt_type default
```

SEE ALSO

set_vt_filler_rule(2)
create_vtcell_fillers(2)
create_stdcell_fillers(2)

set_checkpoint_options

Configure the check pointing system.

SYNTAX

integer **set_checkpoint_options**
[-active *boolean*]
[-runtime_units *runtime_units*]

Data Types

boolean string

ARGUMENTS

-active *boolean*

Enable or disable the checkpointing system. The valid values can be 'true' or 'false'.

-runtime_units *runtime_units*

Sets the units for reporting of runtimes of various checkpoint commands. The valid values can be 'seconds' and 'hours'. By default, the runtime is reported in Hours.

DESCRIPTION

The command configures behaviors of the checkpoint system. The -active option controls whether the checkpoint system is enabled or not. It takes a value of either true or false, defaulting to true.

When the checkpoint system is active:

- eval_checkpoint commands in the flow scripts execute their enwrapped Tcl command/s, and execute all actions or reports associated to that checkpoint
- The ./checkpoint.config.tcl file is sourced during tool startup. This happens after the home init and local init files (like .synopsys_icc2.setup) are sourced, and also after any Tcl commands specified with the -x command line option are executed.
- The ./checkpoint directory is created as soon as any checkpoint related command is encountered (except set_checkpoint_options -active false), or if a ./checkpoint.config.tcl file is found and sourced.
- Checkpoint history of any newly encountered checkpoints is recorded and can be reported with get_checkpoint_data -name,

or by looking at the ./checkpoint/checkpoint_summary.rpt file

Note that when the checkpoint system is active, no actions or reports will be run unless the user first defines and associates them. In terms of impact on the user's flow trajectory and QoR, the active checkpoint system does nothing until such reports and actions are defined and associated. It does still track checkpoint execution history.

When the checkpoint system is inactive:

- eval_checkpoint commands in the flow scripts execute their enwrapped Tcl command/s. Effectively, the Tcl statements enwrapped by the eval_checkpoint command are executed as though they were not wrapped in eval_checkpoint.
- No actions or reports associated with eval_checkpoint are executed
- The ./checkpoint directory will not be created. Note that if a ./checkpoint.config.tcl file exists in the run directory and gets sourced before disabling the checkpoint system, then the ./checkpoint directory will be created.

EXAMPLES

The following example disables the checkpoint system.

```
prompt> set_checkpoint_options -active false
```

The following example sets runtime to be captured in units of 'seconds':

```
prompt> set_checkpoint_options -runtime_units seconds
```

SEE ALSO

create_checkpoint_action(2)
remove_checkpoint_actions(2)
create_checkpoint_report(2)
remove_checkpoint_reports(2)
associate_checkpoint_action(2)
associate_checkpoint_report(2)
get_current_checkpoint(2)
eval_checkpoint(2)
reset_checkpoints(2)
get_checkpoint_data(2)

set_clock_balance_points

Specifies exceptions to apply while balancing clock trees at the specified pins, ports, and clocks.

SYNTAX

string **set_clock_balance_points**
[-rise]
[-fall]
[-early]
[-late]
[-consider_for_balancing true | false]
[-offset *offset delay applied at clock sink pins*]
[-delay *delay_value*]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-clock *clock_list*]
-balance_points *port_pin_list*

Data Types

delay_value float
corner_list list
clock_list list
port_pin_list list

ARGUMENTS

-rise

Specifies that the exception is for the rising edge of the clock at the specified pin or port. If neither **-rise** or **-fall** are specified, then both are assumed to have been applied.

-fall

Specifies that the exception is for the falling edge of the clock at the specified pin or port. If neither **-rise** or **-fall** are specified then both are assumed to have been applied.

-early

Applies the exception for early/shortest path calculations. If neither **-early** or **-late** are specified, then both are assumed to have been applied.

-late

Applies the exception for late/longest path calculations. If neither **-early** or **-late** are specified, then both are assumed to have been applied.

-consider_for_balancing true | false

Specifies whether this constraint excludes a pin or port from balancing. If this option is not provided, then it is assumed that this constraint is for explicitly balancing a point.

When this option is set to true, it becomes a balance-point exception, which is a scenario specific constraint and accepts both **-corner** and **-scenario** options.

When this option is set to false, it becomes an ignore-point exception, which is a mode specific constraint and does not accept either **-corner** or **-scenario** options.

-offset ***offset delay applied at clock sink pins***

Specifies the offset delay which can be applied at clock sink nodes only. This option can not be specified with option **-consider_for_balancing false**.

-delay ***delay_value***

Specifies the delay seen at the pin or port. If this option is specified, **-consider_for_balancing** is assumed to be **true**. It is an error to use **-consider_for_balancing false** and **-delay** together.

-modes ***mode_list***

Specifies the scenarios to which the balance delay value is applied. If this option is given, all scenarios of the specified modes will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners ***corner_list***

Specifies the scenarios to which the balance delay value is applied. If this option is given, all scenarios of the specified corners and the current mode will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios ***scenario_list***

Specifies the scenarios to which the balance delay value is applied. The **-modes** or **-corners** option must not be specified with this option.

-clock ***clock_list***

Specifies a list of clock objects for which to apply the constraint. For clock tree synthesis, generated clocks are a part of the master clock network. Therefore, only non-generated clocks can be specified.

-balance_points ***port_pin_list***

Specifies a list of ports and pins on which to apply the balance point.

DESCRIPTION

This command adds constraints that are used during clock tree synthesis for either excluding or explicitly balancing ports/pins with a specific delay value. A port/pin can only be an exclude point or a balancing point. It cannot be both a balance point for early-rise and an exclude point for late-fall.

Use the **-consider_for_balancing true** option to set balancing constraints on any port or pin on the clock network. If the constraints

are applied to jump pins, then they will be ignored during clock network updates and synthesis. Jump pins are register clock pins that appear in the source latency networks of a generated clock. Constraints for setting exclude pins can be applied on any pin/port on the clock network. Balance point can also be set on hierarchical pins. If hierarchical balance point is connected to a dangling/black box then during **synthesize_clock_trees**, a guide buffer is inserted next to the hierarchical pin and constraint is moved to the input pin of the guide buffer.

Objects marked with only **-consider_for_balancing true** and no **-delay value** option will assume a delay value of zero. If the object and the clock are the same, a new constraint command will overwrite or be combined with a previous constraint. The command issues a warning message if a previous constraint is overwritten. When no early/late values are specified, the corresponding late/early values are copied. The early values cannot be greater than the late values, if such a constraint is given the tool issues a warning and makes the late value equal to the early value.

Balance delay is managed on a per-scenario basis. For designs with multiple scenarios, you can specify different delay values for different scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command applies to the current scenario. The tool issues an error if there is no current scenario. If the **-scenarios** option is specified, the command applies to all of the specified scenarios.

Use the **report_clock_balance_points** command to list the currently specified balance points. **remove_clock_balance_points** command can be used to remove these constraints.

Multicorner-Multimode Support : By default, if **-consider_for_balancing true** is used then this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

If **-consider_for_balancing false** is used then, this command by default is mode specific , and it is applied to all scenarios of a mode . The **-mode** option can be used to apply ignore points exceptions in other modes.

EXAMPLES

The following example sets a balance point at **gck/CP** for rise and early for clock **clk1** with a delay value of 2.0 for the current scenario.

```
prompt> set_clock_balance_points -rise -early -delay 2.0 \
  -clock clk1 -balance_points gck/CP
```

The following example overwrites the previously specified balancing constraint. The command issues a warning message when it overwrites an existing balancing constraint.

```
prompt> set_clock_balance_points -rise -early -delay 2.0 \
  -clock clk1 -balance_points gck/CP
Warning: Object 'gck/CP' had a previously specified constraint set on it for
clock 'clk1', which will be overwritten. (UIC-044)
```

The following example sets an ignore point at **mux/A** for rise and early for clock **clk1**

```
prompt> set_clock_balance_points -consider_for_balancing false \
  -clock clk1 -balance_points mux/A
```

SEE ALSO

remove_clock_balance_points(2)
report_clock_balance_points(2)

set_clock_cell_spacing

Defines spacing rules for the cells in the clock tree.

SYNTAX

status **set_clock_cell_spacing**
[-clocks *clock_list*]
[-lib_cells *libcell_name_list*]
[-x_spacing *x_offset*]
[-y_spacing *y_offset*]

Data Types

clock_list collection
libcell_name_list list
x_offset float
y_offset float

ARGUMENTS

-clocks *clock_list*

Specifies the clock trees for which the cell spacing settings will be applied.

-lib_cells *libcell_name_list*

Specifies a list of library cell names, so that if an instance on the clock tree is of that library cell type, it is affected by specified x,y offset.

-x_spacing *x_offset*

Specifies an offset (spacing) in x_direction. The units are microns.

-y_spacing *y_offset*

Specifies an offset (spacing) in y_direction. The units are microns.

DESCRIPTION

This command addresses a need to apply the placement based design rule (the cell to cell spacing rule) to avoid EM problem on the P/G rails by the cells which are contained in clock networks. The leaf cells (Registers, RAM) of the clock networks will be ignored from this rule. The cells which are not contained in the clock networks will be ignored from this rule. This rule will be applied to the

clock buffers/inverters, the clock gating cells only.

If **-lib_cells** switch is not specified, the rule will be applied to all appropriate cells on the clock network. If **-clocks** option is used then it will apply cell spacing settings for specified clock networks.

With default behavior correct spacing between 2 cells on the clock tree is the maximum of the offsets in each direction. This could be changed to the sum of the offsets in each direction by setting app_option cts.placement.cell_spacing_rule_style to cumulative.

Please note that this command does not directly apply the rules to the relevant instances, it serves only as directive. To apply the rules specified by this command, you need to run the **synthesize_clock_trees** command.

EXAMPLES

The following example specifies clock cell spacing rule for all clock buffers that are of the cell type BUFFD2BWP

```
prompt> set_clock_cell_spacing \
  -x_spacing 0.9 -y_spacing 0.4 -lib_cells mylib/BUFFD2BWP
```

The following example specifies cell spacing rule for all clock cells on the clock tree

```
prompt> set_clock_cell_spacing \
  -x_spacing 0.9 -y_spacing 0.0
```

The following example specifies cell spacing rule for the cells of clk1 clock tree

```
prompt> set_clock_cell_spacing -clocks clk1 \
  -x_spacing 0.9 -y_spacing 0.0
```

SEE ALSO

remove_clock_cell_spacings(2)
report_clock_cell_spacings(2)
cts.placement.cell_spacing_rule_style(3)

set_clock_exclusivity

Specifies a cell for which all the clocks that traverse from input pins to the output pin will be mutually exclusive.

SYNTAX

int **set_clock_exclusivity**
-output output_pin
[-type mux | user_defined]
[-inputs *input_pin_list*]*

ARGUMENTS

-output output_pin

This option specifies a single output pin as the exclusivity point. Depending on the -type option, all or some of the clocks going out of this pin are considered mutually exclusive.

-type mux | user_defined

Specifies the type of clock exclusivity setting, either mux or user_defined. Use the mux setting to set the clock exclusivity for a multiplexer (MUX) cell. Use the user_defined setting for a non-MUX cell.

Use the mux setting only with the -output option, not the -inputs option. The specified output pin must belong to a MUX. All clocks at different data signal inputs of the MUX (inputs that are not MUX select pins) are considered mutually exclusive clocks beyond the output. This behavior is similar to setting the time.enable_auto_mux_clock_exclusivity app option to true.

You can use the user_defined setting for both MUX or non-MUX cells. Use this option with the -inputs and -output options; the specified input and output pins must belong to the same cell in the design. All clocks at the specified inputs are considered mutually exclusive clocks beyond the output.

-inputs input_pin_list

Use this option together with the -type user_defined option. The clocks coming through these input pins are exclusive at the output pin specified by the -output option.

DESCRIPTION

IC designs often contain MUXes in the clock network to select between multiple exclusive clocks. If the MUX select lines are known not to dynamically change, the clocks going through the input pins of the MUX are in fact exclusive.

In this case, to properly constrain a MUXed clock, we can specify the output pin of cell by using set_clock_exclusivity command. The

timing paths that are launched from a clock that goes through one input pin and captured by a clock that goes through another input pin are consid- ered false timing paths. Furthermore, crosstalk interactions between the two clocks coming from different inputs of the exclusive cell are ignored. Please note that if there is a generated clock defined after the exclusivity point, the exclusivity states will be lost. This is similar to the existing behavior of the generated clocks. When a gener- ated clock is created at a pin, all other clocks arriving at that pin are blocked unless they also have generated clock versions created at that pin.

Please note that this feature is not currently supported in extract_model, characterize_context and write_eco_design commands.

To undo the set_clock_exclusivity command, use the remove_clock_exclusivity command.

To report the exclusivities defined in a design, use the report_clock command with the -exclusivity option.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example defines a mux cell output as point of exclusivity.

prompt> **set_clock_exclusivity -type mux -output MUX02/Z**

The following example defines an AND gate as a point of exlusivity.

prompt> **set_clock_exclusivity -type user_defined -output iAND02/Z -inputs {iAND02/A iAND02/B}**

SEE ALSO

remove_clock_exclusivity(2)
report_clock(2)
set_disable_auto_mux_clock_exclusivity(2)
time.enable_auto_mux_clock_exclusivity(3)

set_clock_gate_latency

Specifies clock network latency values to be used for clock-gating cells, as a function of clock domain, clock-gating stage, and transitive fanout.

SYNTAX

status **set_clock_gate_latency**
[-clock *clock_list*]
[-stage *cg_stage*]
[-fanout_latency *cg_fanout_list*]
[-reset]

Data Types

clock_list list
cg_stage integer
cg_fanout_list string

ARGUMENTS

-clock *clock_list*

Specifies that the latency must be applied with respect to the specified clocks. If the **-clock** option is not specified, the latency is applied with respect to all clock domains to which the clock-gating cell belongs. This option cannot be used with the **-reset** option.

-stage *cg_stage*

Specifies the clock-gating stage to which the clock latency data from the fanout range is applied. Registers are considered to be stage 0. Clock gates driving only registers directly or across buffers, inverters, or hierarchical transitions are stage 1. Clock gates, that are stage N, drive at least one clock gate which is stage N-1, directly or across buffers, inverters or hierarchical transitions. The value of N is the maximum possible that can be assigned to a stage. Clock gates whose stage cannot be determined (for example, unloaded clock gates) are assumed to be stage 1, but no value of latency will be applied to an unloaded clock gate.

If this option is used, it must be used with the **-fanout_latency** option. This option cannot be used with **-reset** option.

-fanout_latency *cg_fanout_list*

Specifies the list of latency tuples. When the clock-gating stage is zero (0), the values of each latency tuple are absolute. If the stage is not zero, the values of the latency tuple are relative values. The relative values are calculated with respect to the minimum latency value of the gated cells in the direct fanout of the stage\(aqs clock gate.

Each latency tuple consists of a transitive fanout range and clock latency values. For example, {{1-5 0.9} {6-20 0.5} {21-inf 0.3}}, which consists of a list with 3 latency tuples: a fanout of 1 to 5 has a latency value of 0.9; a fanout of 21 or larger has a latency value of 0.3. If you want the same latency value for the entire fanout range, specify it as {{1-inf 0.9}}. Note that \(dqinf\(dq means

infinity.

If this option is used, it must be used with the **-stage** option. This option cannot be used with the **-reset** option.

The transitive fanout of a clock-gating cell is the set of all clock pins of registers, clock pins of macro cells, and primary output ports which are driven directly or across intermediate objects (such as buffers, inverters, hierarchical transitions, and other clock gates).

-reset

Clears all the current clock gate latency settings that were specified by the **set_clock_gate_latency** command. This option cannot be used with **-clock**, **-stage** or **-fanout_latency** options.

DESCRIPTION

The **set_clock_gate_latency** command allows you to specify clock network latency values for clock-gating cells and registers, as a function of clock domain, clock-gating stage, and transitive fanout. Latency values are calculated and annotated on the clock pins of clock-gating cells and registers during the execution of the **compile** command, or by using the **apply_clock_gate_latency** command. Latency annotation is only supported for flip-flops and clock gates. The tool does not annotate macro cell pins.

When performing annotation on clock pins of clock gate cells, the latency values specified using the **-fanout_latency** option are interpreted as the time required for the clock signal to propagate from the clock-gate cell to the registers or clock gates in its fanout. The clock latency annotated on the clock pin of the gate cell is the clock network latency annotated on its fanout minus the respective relative latency value specified using the **-fanout_latency** option. When performing annotation on registers clock pins, the latency value specified is used as the absolute clock network latency.

Use the **set_clock_gate_latency** command to specify the latency for registers by specifying the command for stage 0. By definition, stage 0 has no fanout, so use the following syntax:

```
prompt> set_clock_gate_latency -stage 0 -fanout_latency \
    {{ 1-inf] value }}
```

When clock latency settings are provided for stage 0, the specified values are annotated on the clock pin of each register, regardless if the register is gated or not. The value that is specified for the given latency tuple is absolute. This means that the specified value is directly annotated on the clock pin of the cell and can be positive or negative

When clock latency settings are provided for nonzero stages, the calculated values are annotated on the clock pin of each clock gate as specified by the command. The annotated value is the annotated latency of on the fanout minus the relative latency specified. The relative latency value must be nonnegative values. If different latency values are detected among the fanout of a clock gate, the lowest latency value is used to compute the relative latency to annotate on the clock gate.

For example, applying the following setting results in registers with an absolute latency value of 1.5. Stage 1 clock gates whose transitive fanout count is lower or equal than 20 get an absolute value of 1.0 (computed as 1.5 - 0.5), and stage 1 clock gates whose transitive fanout count is greater than 20 get an absolute value of 0.8 (computed as 1.5 - 0.7):

```
prompt> set_clock_gate_latency -stage 0 -fanout_latency \
    {{ 1-inf 1.5 }}

prompt> set_clock_gate_latency -stage 1 -fanout_latency \
    {{ 1-20 0.5 } { 21-inf 0.7 }}
```

The tool only annotates the absolute latency value on the clock pin of a cell (clock gate or register) whose stage is N and whose clock domain is **clk**, if the command with the **-stage** N option for that clock domain was explicitly used.

If **set_clock_gate_latency** is used more than one time with the same stage and clock options, the tool overrides the values with the last command given.

The tool can overwrite values that were annotated using optimizations that were enabled by the application option **compile.clockgate.physically_aware**.

When **set_clock_latency** and **set_clock_gate_latency** have different specifications, other rules apply. The following example illustrates the order of precedence:

1) **set_clock_latency -clock clk 0.1 [all_registers -clock_pins]**

2) **set_clock_gate_latency -clock clk -stage 0 -fanout_latency {{1-inf 0.2}}**

3) **set_clock_latency 0.3 [all_registers -clock_pins]**

4) **set_clock_gate_latency -stage 0 -fanout_latency {{1-inf 0.4}}**

5) **set_clock_latency 0.5 [get_clocks]**

The precedence is listed from highest to lowest, regardless of the order in which they are used. All the previous commands can be used independently, in any combination, and in any order. Keep in mind they follow the same precedence as described. If both commands are used with the same specifications, the **set_clock_latency** command prevails over the **set_clock_gate_latency** command.

If you specify fanout ranges in the *cg_fanout_list* and you do not cover all values from 1 to *infinity*, this results in an error. If you specify overlapping ranges in the *cg_fanout_list*, this also results in an error.

If an inconsistent clock latency annotation is detected when running the **apply_clock_gate_latency** or **compile** command, the tool reports a warning message.

If you use the **-reset** option, all the clock gate latency settings that were specified with the **set_clock_gate_latency** command are deleted. Note that this does not remove the annotated latencies that were set during the execution of compile or **apply_clock_gate_latency** command.

To remove annotated latencies that were specified using the **set_clock_gate_latency** command, use the **reset_clock_gate_latency** command.

To report the clock gate latency settings that were specified by the **set_clock_gate_latency** command, use the **report_clock_gate_latency** command.

Multicorner-Multimode Support

This command is scenario dependent and only affects the current scenario. Absolute latency values are calculated with respect to the same scenario.

The actual clock latency annotation is performed during the execution of the compile or **apply_clock_gate_latency** commands, for all the scenarios for which a clock latency value is available.

Latency computation and annotation depends on clock marking which in turn depends on some user constraints like **set_case_analysis** on the clock logic. To guarantee that all clock gates are constrained properly for all clocks, make sure multiple scenarios that cover complementary cases are defined.

Using the **-reset** option deletes all the clock gate latency settings for the current scenario only.

EXAMPLES

The following example specifies the clock latency values for the complete fanout range of clock-gating cells for stages 1, 2, and 3. This latency data applies to the clock-gating cells whose clock pins belong to **clk1**.

```
prompt> set_clock_gate_latency -clock [get_clocks clk1] -stage 1 \
```

```
        -fanout_latency {{1-30 2.1} {31-100 1.7} {101-inf 1.1}}

    prompt> set_clock_gate_latency -clock [get_clocks clk1] -stage 2 \
        -fanout_latency {{1-5 0.9} {6-20 0.5} {21-inf 0.3}}

    prompt> set_clock_gate_latency -clock [get_clocks clk1] -stage 3 \
        -fanout_latency {{1-10 0.28} {11-inf 0.11}}
```

The following example shows how to specify and annotate different clock gate latency values for different multicorner-multimode scenarios.

```
    prompt> current_scenario sc1
    prompt> set_clock_gate_latency -stage 1 \
        -fanout_latency {{1-inf 0.11}}

    prompt> current_scenario sc2
    prompt> set_clock_gate_latency -stage 1 \
        -fanout_latency {{1-inf 0.15}}

    prompt> compile
```

The following example shows how to delete the current clock gate latency settings from the current scenario.

```
    prompt> current_scenario sc1
    prompt> set_clock_gate_latency -reset
```

SEE ALSO

apply_clock_gate_latency(2)
compile(2)
compile.clockgate.physically_aware(3)
remove_clock_latency(2)
report_clock_gate_latency(2)
reset_clock_gate_latency(2)
set_clock_latency(2)

set_clock_gate_routing_rule

Specify a non-default or default routing rule to be applied to the output nets of leaf level tool inserted clock gates during compile.

SYNTAX

status **set_clock_gate_routing_rule**
  [-rule *rule*
    | -default_rule
    | -no_rule
    | -clear]
  [-min_routing_layer *layer*]
  [-max_routing_layer *layer*]
  [-min_layer_mode *mode*]
  [-max_layer_mode *mode*]
  [-min_layer_mode_soft_cost *cost*]
  [-max_layer_mode_soft_cost *cost*]

Data Types

rule	string or collection
layer	list or collection
mode	string
cost	string

ARGUMENTS

-rule *rule*

Specifies the name of the non-default routing rule to be assigned to the leaf level tool inserted clock gate\(qas output nets.

The -rule, -default_rule, -no_rule, and -clear options are mutually exclusive; you can specify only one.

-default_rule

Specifies that the default routing rule should be assigned to the leaf level tool inserted clock gate\(qas output nets.

The -rule, -default_rule, -no_rule, and -clear options are mutually exclusive; you can specify only one.

-no_rule

Specifies no routing rule. This allows the tool to automatically assign a non-default rule or the default rule to the nets.

The -rule, -default_rule, -no_rule, and -clear options are mutually exclusive; you can specify only one.

-clear

Removes the current clock gate routing rule setting, if any. This option will make that no clock gate routing rule setting will be applied during compile. This option is mutually exclusive with all other options.

-min_routing_layer ***layer***

Specifies the minimum routing layer for the nets. Only one layer can be specified.

-max_routing_layer ***layer***

Specifies the maximum routing layer for the nets. Only one layer can be specified.

-min_layer_mode ***mode***

Specifies the minimum routing layer mode for the nets. Valid values are unknown, extract_only, soft, allow_pin_connection, and hard. The default value is unknown.

-max_layer_mode ***mode***

Specifies the maximum routing layer mode for the nets. Valid values are unknown, extract_only, soft, allow_pin_connection, and hard. The default value is unknown.

-min_layer_mode_soft_cost ***cost***

Specifies the minimum routing layer mode soft cost for the nets. Valid values are unknown, low, medium, and high. The default value is unknown.

-max_layer_mode_soft_cost ***cost***

Specifies the maximum routing layer mode soft cost for the nets. Valid values are unknown, low, medium, and high. The default value is unknown.

DESCRIPTION

This command specify a non-default or default routing rule to be applied to the output nets of leaf level tool inserted clock gates during compile. This command also specify minimum and maximum routing layer, mode, and cost constrains to the output nets of leaf level tool inserted clock gates during compile.

Each call to this command will overwrite any previous calls.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies a non-default rule named *cg_ndr* to be set during compile.

```
prompt> set_clock_gate_routing_rule -rule cg_ndr
```

This example clears any settings for this command.

```
prompt> set_clock_gate_routing_rule -clear
```

SEE ALSO

create_routing_rule(2)
remove_routing_rules(2)
report_routing_rules(2)
set_routing_rules(2)

set_clock_gate_style

Sets the types of integrated clock-gating library cells to be used for clock gating insertion.

SYNTAX

status **set_clock_gate_style**
[-test_point before | after | none]
[-observation_output]
[-target *target_list*]
[-objects *object_list*]

Data Types

target_list list
object_list list

ARGUMENTS

-test_point before | after | none

Specifies whether the test control inside the clock gating circuitry is implemented before the latch or after the latch or not at all. If the option is not used, test point *before* is selected by default.

-observation_output

Specifies whether the selected clock gate must have an observability logic output pin. By default selected clock gates don't have this pin.

-target *target_list*

Specifies the target sequential elements to be clock gated by the selected integrated clock-gating library cell type. Valid target values are *pos_edge_flip_flop* and *neg_edge_flip_flop*. If more than one target is included, they must be enclosed in quotes or braces ({}). If the option is not used, both targets *pos_edge_flip_flop* and *neg_edge_flip_flop* are selected by default.

-objects *object_list*

Specifies that the clock gate style should be applied over the given list of design objects. Valid objects are hierarchical instances, power domains and designs. If the option is not used, the clock gate style is set for the top design.

DESCRIPTION

This command specifies the characteristics of integrated clock-gating (ICG) library cells to be used for clock gating insertion.

To be selected for clock gating insertion, an ICG library cell must have the **clock_gating_integrated_cell** attribute set to one of these values:

- latch_posedge
- latch_posedge_precontrol
- latch_posedge_postcontrol
- latch_posedge_precontrol_obs
- latch_posedge_postcontrol_obs
- latch_negedge
- latch_negedge_precontrol
- latch_negedge_postcontrol
- latch_negedge_precontrol_obs
- latch_negedge_postcontrol_obs

In the preceding attribute values the presence of **precontrol** or **postcontrol** substrings means that the test-control logic is located before or after the latch, respectively. This corresponds to the use of **-test_point** option with arguments *before* or *after*, respectively.

The presence of the **obs** substring in the attribute value means that the ICG has observability output pin, and this can be selected by using **-observation_output** option.

Use the **-target** option to specify the type sequential element the current instance of the command applies. This means that the ICG library cell being selected is intended to gate this type of register. Supported targets are:

- Positive-edge triggered flip-flops (use the *pos_edge_flip_flop* argument)
- Negative-edge triggered flip-flops (use the *neg_edge_flip_flop* argument)

Use the **-objects** option to specify different styles for different blocks in the circuit. The clock gate style applies to all cells hierarchically under the specified object, unless a lower-level object has set a different clock gate style. Valid arguments for this option are hierarchical instances, power domains and designs.

The clock gate style is always attached to a netlist object, irrespective of whether the **-objects** option is used or not. When not used, the specified style is attached to the top design. Therefore the execution of the command requires that the design is already loaded in memory, otherwise it errors out. The clock gate style is persistent in the database.

In addition to the **clock_gating_integrated_cell** attribute, **valid_purposes** attribute must be also set on the ICG library cell and its value must contain **cts** purpose. This can be used as a mechanism to allow/disallow the use of specific ICG library cells. To add/remove the **cts** attribute purpose to/from a given ICG library cell, use the **set_lib_cell_purpose** command.

If **set_clock_gate_style** command is not executed at all or the specification for one target is missing, clock gating insertion uses these default values:

- Test control point before the latch.
- No observation logic output pin.

In other words, the tool selects an ICG library cell with attribute value **latch_posedge_precontrol** for positive-edge triggered flip-flops and **latch_negedge_precontrol** for negative-edge triggered flip-flops.

If no latch_negedge_precontrol library cell is available but a latch_posedge_precontrol library cell is available and vice-versa, then the tool uses the ICG library cell that is available. Additional inverters are inferred at the clock input pin of the clock gating cell and between the clock output pin of the clock-gating cell and the clock input pin of the register for functional equivalence.

To check the availability of ICG library cells for a block in the design, use the **check_clock_gate_library_cell_availability** command.

The specified clock gate style is only applicable to the newly instantiated clock gates. The type of cell used for preexisting clock gates, either user-instantiated or inserted by the tool in a previous command, is not modified to fit the current style.

The clock gates are inserted as instances of flat ICG cells, without any hierarchical wrapper around. Discrete clock gates are not supported for insertion, so the technology library must contain ICG cells for the insertion to occur.

The ICG cell circuitry must be latch-based, latch-free clock gating is not supported. The presence of a latch in the clock gating circuitry prevents a leading clock edge by maintaining the value of the clock signal after the trailing edge. For example, for positive-edge triggered flip-flops, the clock signal is forced and kept to logic 0 after the negative edge, preventing a rising edge that loads new data into the register.

Hierarchical instances that have an explicit clock gate style set are not automatically ungrouped. If the ungrouping is explicitly requested by using the **ungroup** command, the clock gate style is lost.

EXAMPLES

The following example specifies that flip-flops with any activation edge have to be clock gated by using ICGs with the test-control circuitry implemented before the latch and without observability output pin. Therefore an ICG with its **clock_gating_integrated_cell** attribute set to **latch_posedge_precontrol** is selected for positive-edge triggered flip-flops, and one with **latch_negedge_precontrol** is selected for negative-edge triggered flip-flops:

```
prompt> set_clock_gate_style -test_point before
The following clock gating style is applied to: top-level (design)
Target: pos_edge_flip_flop neg_edge_flip_flop
Test control position: before
Observation output: false
1
```

The following example specifies that positive-edge triggered flip-flops have to be clock gated by using ICGs with the test-control circuitry implemented after the latch and with observability output pin, while negative-edge triggered flip-flops have to be clock gated by using ICGs without test-control circuitry nor observability output pin. Therefore an ICG with its **clock_gating_integrated_cell** attribute set to **latch_posedge_postcontrol_obs** is selected for positive-edge triggered flip-flops, and one with **latch_negedge** is selected for negative-edge triggered flip-flops:

```
prompt> set_clock_gate_style -test_point after \
  -observation_output \
  -target pos_edge_flip_flop
The following clock gating style is applied to: top-level (design)
Target: pos_edge_flip_flop
Test control position: after
Observation output: true
1

prompt> set_clock_gate_style -test_point none \
  -target neg_edge_flip_flop
The following clock gating style is applied to: top-level (design)
Target: neg_edge_flip_flop
Test control position: none
Observation output: false
1
```

The following example specifies that flip-flops with any activation edge have to be clock gated by using ICGs with the test-control circuitry implemented before the latch. Clock gates must have observability output pin in the whole design, except for hierarchical cells m1/b1 and m2, as well as power domain PD1. So under these specifics objects the selected ICGs have the **clock_gating_integrated_cell** attribute set to either **latch_posedge_precontrol** or **latch_negedge_precontrol**, while for the rest of the circuit the selected ICGs have the attribute set to either **latch_posedge_precontrol_obs** or **latch_negedge_precontrol_obs**.

```
prompt> set_clock_gate_style -test_point before \
  -observation_output
The following clock gating style is applied to: top-level (design)
Target: pos_edge_flip_flop neg_edge_flip_flop
Test control position: before
Observation output: true
1

prompt> set_clock_gate_style -test_point before \
  -objects [get_cells {m1/b1 m2}]
The following clock gating style is applied to: m1/b1 (hier inst) m2 (hier inst)
Target: pos_edge_flip_flop neg_edge_flip_flop
Test control position: before
Observation output: false
1

prompt> set_clock_gate_style -test_point before \
  -objects [get_power_domains PD1]
The following clock gating style is applied to: PD1 (power domain)
Target: pos_edge_flip_flop neg_edge_flip_flop
Test control position: before
Observation output: false
1
```

SEE ALSO

check_clock_gate_library_cell_availability(2)
compile_fusion(2)
report_clock_gating(2)
set_lib_cell_purpose(2)

set_clock_gate_transformations

Sets transformation restrictions on specified clock gates instances during compile.

SYNTAX

status **set_clock_gate_transformations**
object_list true | false
[-split_fanout true | false]
[-ungate_fanout true | false]
[-collapse_levels true | false]
[-merge_equivalents true | false]
[-reset]

Data Types

object_list list

ARGUMENTS

object_list

Specifies the target clock gate instances to apply the transformation restrictions. This option is required unless *-reset* is specified.

true | **false**

Specifies the default for any transformation that it's not explicitly specified by the current command call. This option is required unless *-reset* is specified.

-split_fanout true | **false**

Specifies whether clock gating can move part of a clock gate's fanout to another clone of that clock gate.

-ungate_fanout true | **false**

Specifies whether clock gating can ungate a clock gate's fanout during optimization.

-collapse_levels true | **false**

Specifies whether clock gating can collapse or modify the enable signal of the clock gates during optimization.

-merge_equivalents true | **false**

Specifies whether clock gating can merge equivalent clock gates.

-reset

Removes the previous restrictions set by **set_clock_gate_transformations** on all clock gates. This option cannot be used in

combination with any other option.

DESCRIPTION

Sets restrictions for the *split_fanout*, *ungate_fanout*, *collapse_levels*, and *merge_equivalents* transformations during compile on the specified clock gate instances in *object_list*.

Restricting the *split_fanout* transformation prevents moving part of the clock gate's fanout to another clone of the clock gate. This means that if the clock gate does not honor the maximum fanout, its fanout is not redistributed in other copies of the clock gate. Or, if the clock gate can be collapsed with another upstream clock gate and it also has a side fanout, it cannot be collapsed because that would involve splitting the side fanout.

The *split_fanout* restriction also restricts Physically Aware Clock Gating splitting.

Restricting the *ungate_fanout* transformation prevents the ungating of the clock gate's fanout when trying to honor the minimum bitwidth set by **set_clock_gating_options**. If the clock gate has a constant enable signal or if all its fanout has been removed, the clock gate can still be removed because it is redundant and there is no enable signal transferred to the clock gate's fanout.

Removal of redundant clock gates can be prevented via set_dont_touch or set_size_only. Applying these constraints can also prevent other optimizations.

Restricting the *collapse_levels* transformation prevents the collapse of the clock gate with another upstream or downstream clock gate. This also means that the clock gate's enable function is not modified.

Restricting the *merge_equivalents* transformation prevents the merging of a clock gate with another equivalent clock gate. This restriction does not apply for collapsing with an upstream or downstream clock gates.

A clock gate that have any of the transformation's restriction is selected as the preferred survivor clock gate for the merging and collapsing transformations.

For the merge, collapse, and split transformation, any restriction is inherited by the resulting clock gates.

The restrictions applying to a clock gate can be reported with **report_clock_gating -gating_elements**.

EXAMPLES

The following example restrict the clock gate clock_gate_0 to not be split, but allowing all other transformations.

```
prompt> set_clock_gate_transformations -split_fanout false {clock_gate_0} true
```

The following example allows the clock gate clock_gate_1 to be merged with an equivalent clock gate and ungate its fanout, but restricting all other transformations.

```
prompt> set_clock_gate_transformations -merge_equivalents true -ungate_fanout true {clock_gate_1} false
```

The following example restricts all transformation for all clock gates on the design.

```
prompt> set_clock_gate_transformations [get_clock_gates] false
```

The following example shows how the transformation restrictions are reported in **report_clock_gating -gating_elements**.

```
prompt> set_clock_gate_transformations {clock_gate_2} false
prompt> report_clock_gating -gating_elements
```

```
****************************************
Report : clock_gating
 ...

****************************************


--------------------------------------------------------------------------------
                        Clock Gating Cell Report
--------------------------------------------------------------------------------

 Clock Gating Bank : clock_gate_2 (Level 1) (TLATNTSCAX2MTL) (Pre-Existing)

    Restrictions: dont_ungate_fanout dont_split_fanout dont_merge_equivalents dont_collapse_levels

    INPUTS :
    [CLK  ] clock_gate_2/CK = clk
    [EN   ] clock_gate_2/E = gated_clk1
    [TE   ] clock_gate_2/SE = *Logic0*

    OUTPUTS :
    [ENCLK] clock_gate_2/ECK = gated_clk2
--------------------------------------------------------------------------------


                    Clock Gating Summary
          ------------------------------------------------------------
          |    Number of Clock gating elements        |          1      |
          |                                           |                 |
          |    Number of Tool-Inserted Clock gates    |       0   (0.00%) |
          |                                           |                 |
          |    Number of Pre-Existing Clock gates     |       1 (100.00%) |
          |                                           |                 |
          |    Number of Gated registers              |       8  (80.00%) |
          |                                           |                 |
          |    Number of Tool-Inserted Gated registers |       0   (0.00%) |
          |                                           |                 |
          |    Number of Pre-Existing Gated registers  |       8 (100.00%) |
          |                                           |                 |
          |    Number of Ungated registers            |       0   (0.00%) |
          |                                           |                 |
          |    Total number of registers              |          8      |
          |                                           |                 |
          |    Max number of Clock Gate Levels        |          1      |
          |                                           |                 |
          |    Number of Multi Level Clock Gates      |          0      |
          ------------------------------------------------------------
```

SEE ALSO

report_clock_gating(2)
set_clock_gating_options(2)

set_clock_gating_check

Specifies the value of setup and hold time for clock gating checks.

SYNTAX

string **set_clock_gating_check**
[-setup *setup_value*]
[-hold *hold_value*]
[-rise | -fall]
[-high | -low]
[*object_list*]

float *setup_value*
float *hold_value*
list *object_list*

ARGUMENTS

-setup *setup_value*

Specifies the clock gating setup time. The default is 0.0.

-hold *hold_value*

Specifies the clock gating hold time. The default is 0.0.

-rise

Indicates that only rising delays are to be constrained. By default, if neither **-rise** nor **-fall** are specified, both rising and falling delays are constrained.

-fall

Indicates that only falling delays are to be constrained. By default, if neither **-rise** nor **-fall** are specified, both rising and falling delays are constrained.

-high

Indicate that the check is to be performed on the high level of the clock. By default, the tool determines whether to use the high or low level of the clock using information from the cell's logic. That is, for AND and NAND gates the tool performs the check on the high level; for OR and NOR gates, on the low level. For some complex cells (for example, MUX, OR-AND) the tool cannot determine which to use, and does not perform checks unless you specify either **-high** or **-low**. If the user-specified value differs from that derived by the tool, the user-specified value takes precedence. Unlike setup or hold time, this option sets the attribute only on the specified pin or cell and does not affect the transitive fanout pin or cell. If you specify **-high** or **-low** you must also specify *object_list*; in that case, *object_list* must not contain a clock or a port.

-low

Indicates that the check is to be performed on the low level of the clock. By default, the tool determines whether to use the high or low clock level using information from the cell's logic. That is, for AND and NAND gates the tool performs the check on the high level; for OR and NOR gates, on the low edge. For some complex cells (for example, MUX, OR-AND) the tool cannot determine which to use, and does not perform checks unless you specify either **-high** or **-low**. If the user-specified value differs from that derived by the tool, the user-specified value takes precedence. Unlike setup or hold time, this option sets the attribute only on the specified pin or cell and does not affect the transitive fanout pin or cell. If you specify **-high** or **-low** you must also specify *object_list*; in that case, *object_list* must not contain a clock or a port.

object_list

Specifies a list of objects in the current design for which the clock gating check is to be applied. The objects can be clocks, ports, pins, or cells. If a cell is specified, all input pins of that cell are affected. If a pin, cell, or port is specified, all gates in the transitive fanout are affected. If a clock is specified, the clock gating check is applied to all gating gates driven by that clock. If you specify **-high** or **-low** you must also specify *object_list*; in that case, *object_list* must not contain a clock or a port. By default, if *object_list* is not specified, the clock gating check is applied to the current design.

DESCRIPTION

The **set_clock_gating_check** command specifies a setup or hold time clock gating check to be used for clocks, pins, or cells. The gating check is performed on pins that gate a clock signal.

The clock gating setup check is used to ensure the controlling data signals are stable before the clock is active. This check is performed on combinational gates through which the clock signals are propagated. The arrival time of the leading edge of the clock pin is checked against both levels of any data signals gating the clock. A clock gating setup failure can cause either a glitch at the leading edge of the clock pulse, or a clipped clock pulse.

The clock gating hold check is used to ensure that the controlling data signals are stable while the clock is active. The arrival time of the trailing edge of the clock pin is checked against both levels of any data signal gating the clipped clock pulse.

The options **-high** and **-low** are intended to specify clock gating checks that the tool cannot determine. These options apply only to the pins specified and not to the cells or pins in the transitive fanout. Conversely, setup and hold values for clock gating checks apply to the cells and pins in the transitive fanout. If you need to use the **-setup** or **-hold** options and also the **-high** or **-low** options, issue two separate **set_clock_gating_check** commands to avoid confusion as to what is propagated and what is not propagated.

Use the **remove_clock_gating_check** command to remove information set by **set_clock_gating_check**. The **reset_design** command removes all attributes from the design, including those set by **set_clock_gating_check**.

Use the **report_clock_gating_check** command to report the information for the clock gating check.

When a cell object is given to **set_clock_gating_check** command it is the same as setting it on all the pins of that cell. The precedence for setup and hold values is: enable pin, clock pin, library cell value, clock value, design value.

The **set_clock_gating_check** command does not in fact create a clock-gating check. The correct conditions for a clock-gating check to occur need to already exist on the specified objects.

The **set_clock_gating_check** may be used to modify the properties of an inferred or library-specified clock-gating check.

The command need only be used when either a non-zero setup or hold time check is required or a high or low pulse check is desired.

Solvnet article 015769 "How Are Clock Gating Checks Inferred?" describes clock-gating checks in detail.

Multicorner-Multimode Support

This command works only on the mode associated with the current scenario.

EXAMPLES

The following example specifies a setup time of 0.2 and a hold time of 0.4 for all gates in the clock network of clock CK1.

```
prompt> set_clock_gating_check -setup 0.2 -hold 0.4 [get_clocks CK1]
```

The following example specifies a setup time of 0.5 on the gate and1.

```
prompt> set_clock_gating_check -setup 0.5 [get_cells and1]
```

SEE ALSO

remove_clock_gating_check(2)
current_design(2)
reset_design(2)
report_clock_gating_check(2)
set_disable_clock_gating_check(2)
Solvnet article 015769
time.disable_clock_gating_checks(3)
time.clock_gating_user_setting_only(3)

set_clock_gating_objects

Controls the clock gating for specified objects in the current design and overrides the default behavior for clock gating in **compile**. Objects can be a register, hierarchical cell, power domain, or module.

SYNTAX

status **set_clock_gating_objects**
[-exclude *object_list*]
[-force *object_list*]
[-include *object_list*]
[-clear *object_list*]
[-enable_source *signal_type*]
[-reset]

Data Types

object_list list
signal_type none, enable_pin_only, feedback_loop_only, both, prefer_enable_pin, prefer_feedback_loop

ARGUMENTS

-exclude *object_list*

Specifies a list of objects in the current design to be excluded from clock gating. An object can only appear in one object_list for the following options: **-force**, **-exclude**, **-include** and **-clear**.

-force *object_list*

Specifies a list of objects in the current design to be forcibly gated during clock gate insertion. An object can only appear in one object_list for the following options: **-force**, **-exclude**, **-include** and **-clear**.

-include *object_list*

Specifies a list of objects in the current design for which clock gating should be done as specified by the **set_clock_gate_style** and **set_clock_gating_options** commands. This is the default for all objects in the design. An object can only appear in one object_list for the following options: **-force**, **-exclude**, **-include** and **-clear**.

-clear *object_list*

Specifies a list of objects in the current design for which the inclusion or exclusion criteria for clock gating should be removed. An object can only appear in one object_list for the following options: **-force**, **-exclude**, **-include** and **-clear**.

-enable_source *signal_type*

Specifies the signal source from which the enable logic will be obtained. This option applies to all objects specified for the **-force** and/or **-include** option of the same command instance. When the enable_source option is used the **-clear**, **-exclude**, and **-reset**

option must not be used.

-reset

Removes the criteria previously set by the **set_clock_gating_objects** command. This option cannot be used in combination with any other option.

DESCRIPTION

This command specifies the objects on which clock gating is to be either forced, enabled, or disabled, overriding all conditions necessary for automatic clock gating by the **compile** command.

Use the **-exclude** option to specify that regular clock gating must NOT gate the specified objects.

Use the **-force** option to specify that regular clock gating must gate the specified objects. If the object does not belong to a bank that fulfills the minimum bitwidth requirement, the forced object is gated with an always-enabled clock gate. Note that the tool-inserted always-enabled clock gates are not required to meet the minimum bitwidth. The **-force** option takes precedence over the minimum bitwidth requirement.

Use the **-enable_source** option to control for each register whether to use either only the synch-enable connection (*enable_pin_only*), or only the feedback loop (*feedback_loop_only*), or both feedback loop and synch-enable (default, *both*), or neither synch-enable nor feedback loop (*none*), or feedback loop only if no non-constant synch-enable connection (*prefer_enable_pin*), or synch-enable only if no feedback loop (*prefer_feedack_loop*). The meaning of the signal types is detailed below:

- **-enable_source none** means that no enable condition is derived from any feature of the specified registers. The registers will either not be gated, or if -force is used for them, be gated by an always enable clock gate. The use of **-enable_source none** together with **-include** is an error.
- **-enable_source enable_pin_only** means that only the synchronous enable connection, if any, will be used for computing the enable function for clock gating the specified registers. Note that in case of non-constant synchronous set / synchronous clear signals, the clock will be enabled whenever these are active.
- **-enable_source feedback_loop_only** means that only an enable condition derived from the feedback loop around the register, if any, will be used for computing the enable function for clock gating the specified registers. Note that in case of non-constant synchronous set / synchronous clear signals, the clock will be enabled whenever these are active.
- **-enable_source both** means that any enable condition derived from both the feedback loop, if any, and the synchronous enable connection, if any, of the specified registers will be used for clock gating. Note that in case of non-constant synchronous set / synchronous clear signals, the clock will be enabled whenever these are active. This is the default behavior for all registers.
- **-enable_source prefer_enable_pin** means that, if the specified registers have both a synchronous enable connection and a feedback loop, only the enable condition derived from the enable connection will be used for gating. The feedback loop will only be used, if there is no non-constant enable connection. Note that in case of non-constant synchronous set / synchronous clear signals, the clock will be enabled whenever these are active.
- **-enable_source prefer_feedback_loop** means that, if the specified registers have both a synchronous enable connection and a feedback loop, only the enable condition derived from the feedback loop will be used for gating. The enable connection will only be used, if there is no feedback loop from which an enable condition could be derived. Note that in case of non-constant synchronous set / synchronous clear signals, the clock will be enabled whenever these are active.

Use the **-include** option to specify that regular clock gating criteria should be used on the specified objects.

Use the **-clear** option to remove a previously specified criteria on the specified objects.

For modules and power domains, the criteria is applied to the instances associated with each of the specified objects at the moment the command is executed. Future instances of modules or power domains will not have this criteria.

For an instance referenced by more than one object in one call of the command, the criteria is applied in the following order, overwriting any previously set criteria:

- Set to modules.
- Set to power domains.
- Set to instances.

You must run the **set_clock_gating_objects** command before running the **compile** command. The specifications persist during the subsequent executions of **compile**.

EXAMPLES

The following example excludes the register bank A_reg in the current design, from clock gating:

```
prompt> set_clock_gating_objects -exclude A_reg[*]
```

The following example removes the directive to include and exclude clock gating for the registers the ADDER:

```
prompt> set_clock_gating_objects \
        -clear {ADDER/out1_reg[*] ADDER/out2_reg[*]}
```

The following example excludes every register present in ADDER, except for the out1_reg bank, which is clock gated according to the specified style.

```
prompt> set_clock_gating_objects \
        -exclude ADDER \
        -include ADDER/out1_reg[*]
```

The following example removes the directive to include or exclude clock gating for all objects:

```
prompt> set_clock_gating_objects -reset
```

The following example excludes the module mid and includes mid_A, an instance of mid:

```
prompt> set_clock_gating_objects -exclude mid -include mid_A
```

The following example forces the registers inside the mid_A hierarchy to be gated with always enable clock gates.

```
prompt> set_clock_gating_objects -force {mid_A} -enable_source none
```

The following example specify that the registers inside the mid_A hierarchy should be gated using just the synch enable pin signal. It also forces the tool to insert a clock gate before the register mid_A/out_reg[1] using just the synch enable pin signal or an always enable signal.

```
prompt> set_clock_gating_objects -include {mid_A} -force {mid_A/out_reg[1]} -enable_source enable_pin_only
```

SEE ALSO

compile(2)

set_clock_gate_style(2)
set_clock_gating_options(2)

set_clock_gating_options

Set the structural options for clock gate insertion and configuration in the compile flow.

SYNTAX

status **set_clock_gating_options**
[-minimum_bitwidth *bit_count*]
[-max_fanout *reg_count*]
[-objects *object_list*]

Data Types

bit_count integer
reg_count integer
object_list list

ARGUMENTS

-minimum_bitwidth *bit_count*

Specifies the minimum accumulated bitwidth (See CLOCK GATE FANOUT) that a clock gate can gate. It is an integer value, greater than or equal to one. This setting is optional, and if not set it defaults to 3.

-max_fanout *reg_count*

Specifies the maximum register count (See CLOCK GATE FANOUT) that a clock gate can gate. It is an integer value, greater than or equal to one. This setting is optional, and if not set the fanout of a clock gate is unlimited.

-objects *object_list*

Specifies that the clock gating options are applied to the specified list of design objects. Valid objects are hierarchical instances, power domains and designs. If this option is not used, the clock-gating options are set for the entire design.

DESCRIPTION

This command sets the structural options for clock gate insertion and configuration in the compile flow.

CLOCK GATE FANOUT (BITWIDTH/COUNT)

The clock gate fanout is counted in two ways:

- Accumulated bitwidth, which is the sum of the bitwidths of each register,

either latches or flip-flops, whose clock pin is driven by a clock gate directly or across buffers, inverters or hierarchical transitions.

- Register count, which refers to the number of latches or flip-flops whose clock pin is driven by a clock gate, either directly or across buffers, inverters or hierarchical transitions.

Note that for honoring the minimum bitwidth, only the accumulated bitwidths of the gated flip-flops or latches will be counted. As for the maximum fanout, all pins, either connected directly or through buffers, inverters or hierarchical transitions, will be counted as part of the fanout, including:

- Any pin of a clock gate.
- Any non-clock pin of a flip-flop or a latch.
- Primary outputs.
- Any pin of a combinational cell.
- Any pin of a sequential cell that is neither a flip-flop nor a latch.

MIN AND MAX VALUES

In order to ensure that the tool can create clock-gating banks for single and multibit registers, the minimum bitwidth value is required to be less than or equal to the maximum fanout value.

Value of parameter -minimum_bitwidth is checked several times during the compile flow, by identifying clock gates whose accumulated bitwidth is under the minimum bitwidth requirement. Clock gates driving a small bank will be removed and its fanout ungated unless:

- The clock gate has a constraint that prevents its removal, such as dont_touch or size_only.
- The clock gate is driving two or more clock gates.
- The clock gate is gating an object that cannot be ungated, such as a primary output, a non-clock pin of a flip-flop or latch, etc.

Value of parameter -max_fanout is checked after multibit packing, by identifying clock gates whose fanout exceeds the maximum fanout requirement. The registers that are part of a large clock-gating bank are split into banks using as many clock gates as needed to honor both the minimum bitwidth and max fanout. Using the least amount of clock gates possible, each of them having aproximately the same register fanout. The splitting process does not trigger any ungating of registers.

Note that the tool cannot ensure that both the minimum bitwidth and maximum fanout will be honored after splitting a large clock gating bank if the minimum bitwidth is greater than half of the maximum fanout. In this case, the command issues a WARNING message stating that some clock gates might not honor the max_fanout setting.

If large bank splitting cannot honor the maximum fanout, the tool will evenly distribute the excess of registers among all the clock gates derived from the splitting.

Please note that some of the physical optimizations could lead to small changes in the register fanout of a clock gate, causing the tool to fail to honor the minimum bitwidth or the maximum fanout at the end of compile.

EXAMPLES

The following examples specify the clock-gating options described below:

```
prompt> set_clock_gating_options -max_fanout 3
The following clock gating options are applied to: top-level (design)
Minimum bitwidth: 3
Maximum fanout: 3
```

```
1

prompt> set_clock_gating_options -minimum_bitwidth 6 \
                        -max_fanout 8 \
The following clock gating options are applied to: top-level (design)
Minimum bitwidth: 6
Maximum fanout: 8
1

prompt> set_clock_gating_options -minimum_bitwidth 4 \
                        -max_fanout 6 \
                        -objects [get_cells] \
The following clock gating options are applied to: b1 (hier inst)
Minimum bitwidth: 4
Maximum fanout: 6
1
```

SEE ALSO

compile(2)
report_clock_gating(2)
set_clock_gate_style(2)

set_clock_groups

Specifies clock groups that are mutually exclusive or asynchronous with each other in a design so that the paths between these clocks are not considered during the timing analysis.

SYNTAX

Boolean **set_clock_groups**
[-physically_exclusive | -logically_exclusive | -asynchronous]
[-allow_paths]
[-name *name*]
[-group *clock_list*]*

list *clock_list*

ARGUMENTS

Note: Options marked with asterisks (*) in the SYNTAX can be used more than once in the same command.

-physically_exclusive

Specifies that the clock groups are physically exclusive with each other. Physically exclusive clocks cannot co-exist in the design physically. An example of this is multiple clocks that are defined on the same source pin. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-logically_exclusive

Logically exclusive clocks do not have any functional paths between them, but may have coupling interactions with each other. An example of logically-exclusive clocks is multiple clocks, which are selected by a MUX but can still interact through coupling upstream of the MUX cell. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-asynchronous

Specifies that the clock groups are asynchronous to each other. Two clocks are asynchronous with respect to each other if they have no phase relationship at all. Signal integrity analysis uses an infinite arrival window on the aggressor unless all the arrival windows on the victim net and the aggressor net are defined by synchronous clocks. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-allow_paths

Enable the timing analysis between specified asynchronous clock groups. The default behavior is to suppress timing paths between asynchronous clocks. The *-allow_paths* option must be used with *-asynchronous* option.

-name *name*

Specifies a name for the clock grouping to be created. Each command should specify a unique name, which identifies the

exclusive or asynchronous relationship among specified clock groups, and this name is used later on to easily remove the clock grouping defined here. By default, the command creates a unique name.

-group *clock_list*

Specifies a list of clocks. You can use the **-group** option more than once in a single command execution. Each **-group** iteration specifies a group of clocks, which are exclusive or asynchronous with the clocks in all other groups. If only one group is specified, it means that the clocks in that group are exclusive or asynchronous with all other clocks in the design. Thus, a default other group is created for this single group. Whenever a new clock is created, it is automatically included in the default exclusive or asynchronous group. Substitute the list you want for *clock_list*.

-comment *comment_string*

Associate a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

DESCRIPTION

Specifies clock groups that are mutually exclusive or asynchronous with each other in a design so that the paths between these clocks are not considered during the timing analysis. This is similar to using the set_false_path command. In addition, these relationships also imply the type of crosstalk analysis which should be performed between the clocks. A clock cannot be present in multiple groups during one **set_clock_groups** command execution, but can be included in multiple groups by executing multiple **set_clock_groups** commands. Clock relationships are only implied across the specified clock groups; no relationship is implied across clocks within a group.

For all three relationships, timing paths between the clocks are suppressed. The relationships differ in how crosstalk is handled. If clocks are asynchronous, the clocks are assumed to have an infinite window relationship with each other. If clocks are physically exclusive, only one clock can act as an aggressor or victim during crosstalk analysis; interactions between the clocks will not be explored. If clocks are logically exclusive, the crosstalk computation will be computed normally (synchronously if the clocks are synchronous) even though the timing paths between the clocks are not reported.

Multiple relationship types can exist between two clocks. When this happens, the relationships obey the following precedence:

** physically exclusive (highest)*
** asynchronous*
** logically exclusive (lowest)*

These precedence rules reflect the real physical behavior of the resulting circuit. For example, if two clocks are asynchronous but they are never simultaneously present, then no crosstalk should be computed.

Note that the traditional set_false_path exception between clocks will not result in infinite window crosstalk computation between two clocks. If multiple clocks are asynchronous with each other, the asynchronous relationship should be specified with set_clock_groups. Failure to specify this relationship for asynchronous clocks could result in an optimistic analysis!

A special -allow_paths modifier can be used with the -asynchronous option. When used, the clocks are assumed to have an infinite window relationship for crosstalk purposes, but timing paths between the clocks will be treated normally. When using this option, paths between the clock domains can be manually disabled using the set_false_path command.

When a clock pair is defined such that it has false paths (using either physically or logically exclusive or asynchronous) but subsequently modified to have -asynchronous -allow_paths, then this is an error and the first definition holds. In the reverse situation, also the redefinition to false paths from -allow-paths between the clock pairs would also be an error. Please refer message UITE-457 and UITE-458.

You should not also set a false path if you have already specified two clocks as exclusive or asynchronous. An error message is issued if you attempt to set a false path between two clocks which are already exclusive or asynchronous. If a false path was previously set between two clocks when an exclusive or asynchronous relationship is applied, the false path is overwritten by the

set_clock_groups command. Other exceptions are unaffected.

A clock relationship on a master clock will not automatically propagate to any generated clocks. If the relationship should also apply to generated clocks, they should be explicitly included in the set_clock_groups command.

To undo the **set_clock_groups** command, use the **remove_clock_groups** command. To report the clock groups defined in a design, use the **report_clock** command with the **-groups** option.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example defines two asynchronous clock domains.

```
prompt> set_clock_groups -asynchronous -name g1 -group CLK33 -group CLK50
```

The following example defines a clock named *TESTCLK* to be asynchronous with all other clocks in the design:

```
prompt> set_clock_groups -asynchronous -group TESTCLK
```

The following example shows how to simultaneously analyze multiple clocks per register without manually specifying false paths. Assume two pairs of mutually-exclusive clocks that are multiplexed:

CLK1 and *CLK2*
CLK3 and *CLK4*.

If each pair of clocks is selected by a different signal, you must execute two **set_clock_groups** commands to specify two independent exclusivity relationships, one for each MUX selection signal:

```
prompt> set_clock_groups -physically_exclusive -group CLK1 -group CLK2
prompt> set_clock_groups -physically_exclusive -group CLK3 -group CLK4
```

If each pair of clocks is selected by a single MUX selection signal, you would execute only one **set_clock_groups** command to simultaneously analyze all four clocks.

```
prompt> set_clock_groups -physically_exclusive -group {CLK1 CLK3} -group {CLK2 CLK4}
```

In this case, we are not implying any type of relationship between CLK1-CLK3 or CLK2-CLK4. For example, f CLK1 and CLK3 were also asynchronous with each other, we would need to specify an additional relationship between them.

SEE ALSO

remove_clock_groups(2)
report_clock(2)
set_false_path(2)
create_clock(2)
create_generated_clock(2)

set_clock_jitter

Sets the duty-cycle jitter and/or the cycle clock jitter.

SYNTAX

status **set_clock_jitter**
-clock *clock_list*
[-cycle *cycle_to_cycle_jitter*]
[-duty_cycle *duty_cycle_jitter*]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]

Data Types

clock_list list
cycle_to_cycle_jitter float
duty_cycle_jitter float

ARGUMENTS

-clock *clock_list*

Specifies a list of clock on which to set the clock jitter.

-cycle *cycle_to_cycle_jitter*

Specifies the cycle-to-cycle jitter on the clock.

-duty_cycle *duty_cycle_jitter*

Specifies the duty-cycle jitter on the clock.

-modes *mode_list*

Specifies the scenarios to which the path margin delay value is applied. If this option is given, all scenarios of the given modes are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-corners *corner_list*

Specifies the scenarios to which the path margin delay value is applied. If this option is given, all scenarios of the given corners and the current mode is used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-scenarios *scenario_list*

Specifies the scenarios to which the path margin delay value is applied. The **-modes** or **-corners** option must not be given with this option.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

DESCRIPTION

This command specifies the cycle-to-cycle jitter and the duty-cycle jitter on a primary master clock(a clock that is not a generated clock itself), or a generated clock that is defined with the **-multiply_by** option.

Cycle-to-cycle jitter models the variation between the edges that are in-phase. In other words, it models the variation between the edges that are multiple clock cycles apart.

Duty-cycle jitter models the variation between the edges that are out-of-phase. In other words, they are 0.5, 1.5, 2.5, and so on, cycles apart.

Duty-cycle jitter and cycle-to-cycle jitter are realized between the launch clock and the capture clock that are both generated from the same master clock. To figure out the right clock jitter value between the launch clock and capture clock, we trace the launch edge and the capture edge to corresponding edges of the master clock. There are three cases to consider:

1. If the corresponding edges of the master clock are multiple cycles apart. cycle-to-cycle jitter is realized.
2. If they are not and integral number of cycles apart. duty-cycle jitter is realized.
3. If they are traced back to the same edge of the master clock. no clock jitter is realized.

In crosstalk analysis, clock jitter does not change the arrival windows used in overlap analysis. However, in level-sensitive latch designs, latches can start borrowing in the presence of clock jitter, leading to changes in delta delays for logic nets in the fanout of affected latches.

Please note that the generated clocks defined with the **-multiply_by** option do not inherit the clock jitter from their master clocks.

To remove the clock jitters set by the **set_clock_jitter** command, use the **remove_clock_jitter** command.

To view clock jitter information, use the **report_clock_jitter** command.

EXAMPLES

The following example specifies a cycle-to-cycle jitter of 0.5 and duty-cycle jitter of 0.7 on the clock mclk

prompt> **set_clock_jitter -clock [get_clocks mclk] -cycle 0.5 -duty_cycle 0.7**

SEE ALSO

remove_clock_jitter(2)

report_clock_jitter(2)

set_clock_latency

Specifies the latency of a clock network.

SYNTAX

string **set_clock_latency**
[-clock *clock_list*]
[-rise]
[-fall]
[-min]
[-max]
[-source]
[-offset]
[-late]
[-early]
[-dynamic *dynamic_component_of_delay*]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
delay
object_list

Data Types

clock_list	list
dynamic_component_of_delay	float
mode_list	list
corner_list	list
scenario_list	list
delay	float
object_list	list

ARGUMENTS

-clock *clock_list*

Specifies a list of clock objects to be associated with the latency that is placed on all pin/port objects in *object_list*. If the **-clock** option is supplied when *object_list* refers to clock objects, a warning is issued that the option is not relevant in this case and execution of the command proceeds as if **-clock** was not specified. A more specific clock latency setting overrides a more general one.

-rise

Specifies the clock rise latency.

-fall

Specifies the clock fall latency.

-min

Specifies the clock latency for the minimum operating condition.

-max

Specifies the clock latency for the maximum operating condition.

-source

Specifies the clock source latency. The **-offset** option must not be specified with this option.

-offset

Specifies the offset latency at clock network sink or clock pin of ICG cells. The **-source** option must not be specified with this option.

-late

Specifies the clock late latency.

-early

Specifies the clock early latency.

-dynamic ***dynamic_component_of_delay***

Specifies the dynamic component of clock source latency value.

-modes ***mode_list***

Specifies the scenarios to which the latency value is applied. If this option is given, all scenarios of the specified modes will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners ***corner_list***

Specifies the scenarios to which the latency value is applied. If this option is given, all scenarios of the specified corners and the current mode will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios ***scenario_list***

Specifies the scenarios to which the latency value is applied. The **-modes** or **-corners** option must not be specified with this option.

delay

Specifies the clock latency value.

object_list

Specifies the clocks, ports, or pins.

DESCRIPTION

This command specifies clock latency for a design. Two types of clock latency can be specified for a design: clock network latency and clock source latency.

The clock network latency is the time required for a clock signal to propagate from the clock definition point to a register clock pin. The rise and fall latencies are the latencies for rising and falling transitions at the register clock pin, respectively. Inversion of the clock waveform, if present in the clock network, is not taken into consideration when computing clock network latencies at register clock pins. Note that this behavior is different than it was in previous releases.

Clock source latency is the time required for a clock signal to propagate from its actual ideal waveform origin point to the clock definition point in the design. It can be used to model off-chip clock latency when the clock generation circuit is not part of the current design. You can use clock source latency for generated clocks to model the delay from master-clock to generated-clock definition point.

Dynamic clock latency is the component of the total clock source latency which is considered 'dynamic' in nature. A 'dynamic' value can differ along successive clock cycles and can only be used to calculate CRP if a zero-cycle path is being considered. A zero-cycle path is one in which the same clock edge both launches and captures.

The dynamic component of any clock latency can be specified using the **-dynamic** switch. It can only be specified if the **-source** switch and total clock latency is also specified with the command. Source latency, with dynamic component, can be specified for clocks and clock sources(pins/ports). For clock sources, it must be specified with a relative clock using **-clock** option.

The tool assumes ideal clocking, which means clocks have a specified network latency designated by the **set_clock_latency** command, or zero network latency, by default. Propagated clock network latency, designated by the **set_propagated_clock** command, is normally used for post-layout after final clock tree generation. Ideal clock network latency provides an estimate of the clock tree for pre-layout.

You can specify clock source latency for ideal or propagated clocks. The total clock latency at a register clock pin is the sum of clock source latency and clock network latency. If the rise and fall latencies are different, then the source latencies for a latch are picked based on the sense at the clock port and network latencies are picked based on the sense at the latch clock pin. Thus, source latency takes into account the clock inversions on the clock network, but network latency does not.

In the on-chip analysis mode, the min/max operating conditions are the variation bounds within the single corner analysis. Therefore, only the min-early and max-late latencies are used for the analysis. In the on_chip_variation mode, it is sufficient to specify either **-early**/**-late** options or **-min**/**-max** options for **set_clock_latency**.

Clock latency is managed on a per-scenario basis. For designs with multiple scenarios, you can specify different latency values for different scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command applies to the current scenario. The tool issues an error if there is no current scenario. If the **-scenarios** option is specified, the command applies to all of the specified scenarios.

If the **-modes** option is specified, the command applies to all scenarios of the specified modes. The tool issues an error if none of the specified modes have any scenarios.

If the **-corners** option is specified, the command applies to all scenarios of the specified corners and the current mode. The tool issues an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are specified, the command applies to all scenarios of the specified corners and the specified modes. The tool issues an error if there are no such scenarios. It is an error to specify **-scenarios** with **-modes** or **-corners**.

For internally-generated clocks, the tool automatically computes the clock source latency provided that the master clock of the generated clock has propagated latency and no user-specified value for the generated clock source latency exists. If the master clock is ideal, the master clock has source latency, and there is no user-specified value for the generated clock's source latency, then zero source latency is assumed.

If the **set_clock_latency** command is applied to pins or ports, it affects all register clock pins in the transitive fanout of the pins or ports. Directly setting a network latency makes the affected registers ideal. A warning is issued if the **set_propagated_clock** command has previously set a propagated attribute on the same clock, pin, or port object. Clock source latencies are allowed only on clocks and clock source pins.

No check will be performed at the UI level that a clock specified with **-clock** passes through the pin or pins referenced in object_list

or not. If the clock specified does not pass through the pin, the command will have no effect but no warning of this fact is given.

To undo **set_clock_latency**, use the **remove_clock_latency** command.

To see clock network and source latency information (including dynamic component), use the **report_clocks** command with the **-skew** option.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example specifies a rise latency of **1.2** and a fall latency of **0.9** for a clock named **CLK1**.

```
prompt> set_clock_latency 1.2 \
  -rise [get_clocks CLK1]
prompt> set_clock_latency 0.9 \
  -fall [get_clocks CLK1]
```

The following example specifies an early rise and a fall source latency of **0.8** and a late rise and fall source latency of **0.9** for a clock named **CLK1**.

```
prompt> set_clock_latency 0.8 -source \
  -early [get_clocks CLK1]
prompt> set_clock_latency 0.9 -source \
  -late [get_clocks CLK1]
```

The following example specifies an early and late source latency of 3 and 5 respectively with dynamic components of 0.5.

```
prompt> set_clock_latency 3 -source \
  -early -dynamic 0.5 [get_clocks CLK1]
prompt> set_clock_latency 5 -source \
  -late -dynamic 0.5 [get_clocks CLK1]
```

The following example specifies an early and late source latency of 3 and 5 respectively with dynamic components of 0.5 on clock source port CLK with relative clock CLK1.

```
prompt> set_clock_latency 3 -source \
  -early -dynamic 0.5 -clock [get_clocks CLK1] [get_ports CLK]
prompt> set_clock_latency 5 -source \
  -late -dynamic 0.5 -clock [get_clocks CLK1] [get_ports CLK]
```

The following example specifies the source and the network latencies for three corners of the current mode: corner1, corner2 and corner3. The source latency for CLK1 for corner1 is set to 4 and network latency to 1.

```
prompt> set_clock_latency 4 -source \
  [get_clocks CLK1] -corners [get_corners {corner1}]
prompt> set_clock_latency 5 -source \
  [get_clocks CLK1] -corners [get_corners {corner2 corner3}]
prompt> set_clock_latency 1 [get_clocks CLK1]
prompt> set_clock_latency 2 [get_clocks CLK1] -corners [get_corners {corner2}]
prompt> set_clock_latency 3 [get_clocks CLK1] -corners [get_corners {corner3}]
```

SEE ALSO

remove_clock_latency(2)
report_clocks(2)
set_clock_latency(2)
set_clock_transition(2)
set_clock_uncertainty(2)
set_propagated_clock(2)

set_clock_routing_rules

Specifies routing rules used in clock tree synthesis.

SYNTAX

status **set_clock_routing_rules**
[-clocks *clock_list*]
[-nets *net_list*]
[-net_type all | sink | internal | root]
[-min_routing_layer *min_layer*]
[-max_routing_layer *max_layer*]
[-rules *ndr_rule*]
[-default_rule]

Data Types

clock_list collection
net_list collection
min_layer string
max_layer string
ndr_rule string

ARGUMENTS

-clocks *clock_list*

Specifies the clock trees on which to apply the routing rule. You cannot specify generated clocks.

If you do not specify this option, the rule is applied to all clock trees in the design.

This option and the **-nets** option are mutually exclusive.

-nets *net_list*

Specifies the nets on which to apply the routing rule. The rules are propagated through the derived nets.

This option and the **-clocks** option are mutually exclusive.

-net_type *all* | *sink* | *internal* | *root*

Specifies the type of nets on which to apply the routing rule. The valid values are

- **sink** - the nets that connect to one or more clock leaf pins
- **root** - the nets that are driven by the clock source When synthesis inserts buffers, this type applies to the buffer chain before branching out.

- **internal** - the rest of the nets in the clock tree
- **all** (the default) - all nets

This option cannot be used with the **-nets** option.

-min_routing_layer ***min_layer***

Specifies the minimum routing layer for this clock routing rule. Specify the routing layer by using the layer names from the technology file; you can specify only one layer.

-max_routing_layer ***max_layer***

Specifies the maximum routing layer for this clock routing rule. Specify the routing layer by using the layer names from the technology file; you can specify only one layer.

-rules ***ndr_rule***

Specifies a nondefault routing rule used in clock tree synthesis. To define a nondefault routing rule, use the **create_routing_rule** command.

-default_rule

Applies the default routing rule to the clock nets.

DESCRIPTION

This command assigns the routing rules used in clock tree synthesis. You can associate a rule with specific clocks or nets. If a net already has a nondefault routing rule, this rule is also honored.

EXAMPLES

The following example assigns a nondefault rule named dblespacing to the sink-level nets of the clk1 clock.

```
prompt> set_clock_routing_rules -clocks clk1 -net_type sink \
  -rules dblespacing
```

SEE ALSO

create_routing_rule(2)
remove_clock_routing_rules(2)
remove_routing_rules(2)
report_clock_routing_rules(2)
report_routing_rules(2)
set_routing_rule(2)

set_clock_sense

The set_clock_sense command is deprecated. Use the set_sense command instead.

Multicorner-Multimode Support

This command works only on the current mode.

SEE ALSO

set_sense(2)
remove_sense(2)

set_clock_transition

Specifies the transition time for register clock pins.

SYNTAX

string **set_clock_transition** [-rise]
[-fall]
[-min]
[-max]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
transition
clock_list

mode_list list
corner_list list
scenario_list list
transition float
clock_list list

ARGUMENTS

-rise

Specifies clock transition time for rising clock edge.

-fall

Specifies clock transition time for falling clock edge.

-min

Specifies clock transition time for minimum conditions.

-max

Specifies clock transition time for maximum conditions.

-modes *mode_list*

Specifies the scenarios to which the transition time is applied. If this option is given, all scenarios of the specified modes are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by -corners are used.

-corners *corner_list*

Specifies the scenarios to which the transition time is applied. If this option is given, all scenarios of the specified corners and the current mode are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by -corners are used.

-scenarios ***scenario_list***

Specifies the scenarios to which the transition time is applied. The **-modes** or **-corners** option must not be specified with this option.

transition

Specifies transition time of clock pins.

clock_list

Provides a list of clocks in the design. The transition times on all register clock pins in the transitive fanout of specified clock are affected. You can only specify clocks with ideal latency in the list. When register clock pins in the fanout of a clock are marked with propagated latency, **set_clock_transition** values are ignored.

DESCRIPTION

This command overrides the calculated transition times on clock pins of registers.

This command is especially useful for pre-layout when clock trees are incomplete and calculated transition times at register clock pins can be highly pessimistic. The transition value specified with this command overrides the transition times of all nets directly feeding a sequential element clocked by the specified clock.

Use the **set_clock_transition** command only with ideal clocks. For propagated clocks, the calculated transition times are used. Propagated clocks are only set after the final clock tree is constructed.

If a clock transition is not specified for an ideal clock, and if the **time.ideal_clock_zero_default_transition** application option is **true** (the default), zero transition is used. Otherwise, the transition time is calculated as it is for other pins in the design.

Clock transition data is managed on a per-scenario basis. For designs with multiple scenarios, you can specify different transitions for different scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the specified transition is applied to the current scenario. The command issues an error if there is no current scenario. If the **-scenarios** option is specified, the transition is applied to all of the specified scenarios. If the **-modes** option is specified, the transition will be applied to all scenarios of the specified modes. The command issues an error if none of the specified modes have any scenarios. If the **-corners** option is specified, the transition will be applied to all scenarios of the specified corners and the current mode. The command issues an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are specified, the transition is applied to all scenarios of the specified corners and the specified modes. The command issues an error if there are no such scenarios. It is an error to give -scenarios with **-modes** or **-corners**.

To undo **set_clock_transition**, use **remove_clock_transition**.

To list all clock transition values which have been set, use **report_clocks -skew**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

This example specifies a rise transition time of 0.38 and fall transition time of 0.25 for register clock pins clocked by "CLK1".

```
prompt> set_clock_transition 0.38 -rise [get_clocks CLK1]
prompt> set_clock_transition 0.25 -fall [get_clocks CLK1]
prompt> set_clock_transition 0.35 -corners [get_corners C1] [get_clocks CLK1]
prompt> set_clock_transition 0.45 -fall -corners [get_corners C2] [get_clocks CLK1]
```

SEE ALSO

remove_clock_transition(2)
report_clocks(2)
set_clock_latency(2)
set_clock_uncertainty(2)
set_propagated_clock(2)
timing_ideal_clock_zero_default_transition(3)

set_clock_tree_options

Set target skew/latency constraints for clock trees.

SYNTAX

status **set_clock_tree_options**
[-clocks *clock_list*]
[-corners *corner_list*]
[-target_skew *skew_value*]
[-target_latency *latency_value*]
[-copy_exceptions_across_modes]
[-from_mode *original_mode*]
[-to_mode *mode_list*]
[-root_ndr_fanout_limit *fanout_limit_value*]
[-max_incr_repeater_levels *level_value*]]

Data Types

clock_list list or collection
corner_list list or collection
mode_list list or collection
original_mode object
skew_value float
latency_value float
fanout_limit_value integer
level_value integer

ARGUMENTS

-clocks *clock_list*

Specifies the clock trees to which target skew/latency is applied. If not specified, the constraints are applied to all clocks.

-corners *corner_list*

Specifies the corners to which target skew/latency is applied. If not specified, the constraints are applied to current corner.

-target_skew *skew_value*

Specifies the required value for maximum skew for the specified clock trees. The default value for this option is 0.

-target_latency *latency_value*

Specifies the minimum early insertion delay constraint for the specified clock trees. There is no default value for this option.

-copy_exceptions_across_modes

Enables the automatic copying of CTS balance points (considered for balancing) settings from the mode specified by the **-from_mode** option to the mode specified by the **-to_mode** option.

You must specify the **-copy_exceptions_across_modes**, **-from_mode**, and **-to_mode** options together.

You can specify only one group of **-from_mode** and **-to_mode** settings. If additional groups are needed, they need additional commands. The modes across different groups must be mutually exclusive.

The same corner that the balance point was defined on must be present in the mode specified by the **-to_mode** option for the copy to be successful.

-from_mode ***original_mode***

Specifies the mode from which to copy to CTS balance points. You can specify only one mode.

You must specify the **-copy_exceptions_across_modes**, **-from_mode**, and **-to_mode** options together.

-to_mode ***mode_list***

Specifies the modes to which to copy the CTS balance points. You can specify one or more modes.

You must specify the **-copy_exceptions_across_modes**, **-from_mode**, and **-to_mode** options together.

-root_ndr_fanout_limit ***fanout_limit_value***

Specifies the boundary on the clock nets where root NDR/layer or internal NDR/layer should be applied, depending on the number of transitive fanout of the clock net driver. The transitive fanouts include register CK pin that are valid sinks and does not include ignore pins and they are counted per master clock. If a net driver is on multiple (master) clock domain, the max of downstream transitive fanouts for each clock domain will be considered. If the downstream transitive fanout of a net driver is larger than or equal to this user-specified fanout limit, root NDR/layer would be used for that net; If the downstream transitive fanout of a net driver is smaller than this user-specified fanout limit, internal NDR/layer would be used for that net; This is also called fanout-based ndr.

-max_incr_repeater_levels ***level_value***

Specifies the maximum number of incrementally added repeater level during concurrent clock and data (CCD) optimization, depending on the repeater level of each sink. The **-clock** option could be used together to specify clock. Newer setting will overwrite the older one on the same clock and be treated the same as the first time it's being set.

DESCRIPTION

Specifies settings like target skew/latency for clocks, exception duplication across modes or fanout-based ndr limit.

If target skew is specified, clock tree synthesis engine minimizes skew to meet the specified target. After this target is met, the optimization concentrates more on other QoR goals, such as insertion delay and area.

If target latency is specified, clock tree synthesis engine minimizes latency to meet the specified target. After this target is met, the optimization concentrates more on other QoR goals, such as skew and area. If the insertion delay of this initial optimized clock tree is smaller than the specified value, the clock tree synthesis engine adds a chain of cells from the reference list as needed to meet this delay.

If fanout-based ndr is specified, clock tree synthesis engine will apply root/internal rules based on the limit setting. The sink ndr rule will not be affected and remain as sink ndr rule. Small net (e.g., bbox < 10um) will still have internal rules even if its fanout is larger than limit.

To report target skew/latency constraint settings, use **report_clock_tree_options**.

To remove target skew/latency constraints, use **remove_clock_tree_options**.

To report fanout-based ndr settings, use **report_clock_tree_options**.

To remove fanout-based ndr settings, use **remove_clock_tree_options**.

Multicorner-Multimode Support

EXAMPLES

The following example sets a target skew of 2.0 units for clock clk1.

prompt> **set_clock_tree_options -target_skew 2.0 -clock clk1**

The following example sets a target latency of 4.0 units for all clocks.

prompt> **set_clock_tree_options -target_latency 4.0**

SEE ALSO

report_clock_tree_options(2)
remove_clock_tree_options(2)

set_clock_tree_reference_subset

Specifies reference cells to be used in clock tree synthesis.

SYNTAX

status **set_clock_tree_reference_subset**
[-clocks *clock_list*]
[-lib_cells *references*]

Data Types

clock_list collection
references collection

ARGUMENTS

-clocks *clock_list*

Specifies the clock trees for which the lib_cells will be applied. A generated clock is not accepted in this command. This option is mandatory with **-lib_cells** option.

-lib_cells *references*

Specifies the list of references to be used in clock trees.

DESCRIPTION

The command specifies buffers, inverters, and clock gates to be used for clock tree synthesis. Specifying several references allows the clock tree synthesis engine more flexibility to build clock trees and provides improved clock tree results.

Clock tree synthesis will pick up lib_cells from intersection of **set_lib_cell_purpose -include cts <lib-cells>** and **set_clock_tree_reference_subset**.

You can use the **-clocks** option to apply reference lists on a per-clock basis. You can specify clock gates as reference cells during optimization for cell sizing. To size a gate, equivalent gates from reference lists are used.

If you issue the **set_clock_tree_references_subset** command several times for the same clock tree, the new references you specify will over write existing references for that clock tree. You can view the references of a given clock tree by using the **report_clock_tree_reference_subset** command. To remove references, use the **remove_clock_tree_reference_subset** command.

Clock tree synthesis will pick up union of lib_cells for overlapping clocks with clock-specific reference_subsets.

EXAMPLES

The following example specifies buf1 lib cell for clock tree synthesis for clock tree clk1.

prompt> **set_clock_tree_references_subset -clocks clk1 -lib_cells buf1**

SEE ALSO

set_lib_cell_purpose(2)
report_clock_tree_reference_subset(2)
remove_clock_tree_reference_subset(2)

set_clock_trunk_endpoints

Specifies pins or ports as clock trunk endpoints for clock trunk planning.

SYNTAX

status **set_clock_trunk_endpoints**
[-clock *clocks*]
[-corners *corners*]
[-delay *delay_behind_endpoint*]
[-auto_clock connected | local | all]
pins_or_ports

Data Types

clocks	collection
corners	list
delay_behind_endpoint	float
pins_or_ports	collection

ARGUMENTS

-clock *clocks*

Specifies the clocks for which to create specific endpoints. By default the specified endpoints apply to all clocks.

-corners *corners*

Specifies the corners for which to create specific endpoints. By default, the specified endpoints apply to all corners.

-delay *delay_behind_endpoint*

Specifies the phase delay from the endpoint to the clock sinks which should be assumed for clock trunk planning. If no phase delay is given, the phase delay is undefined and clock trunk planning estimates the phase delay.

pins_or_ports

Specifies the pins or ports to define as a clock trunk endpoint.

-auto_clock connected | local | all

Associates clocks at the block level with clocks at the top level. For example, your top-level clock might be called "SYS_CLK", and the same clock in the block might be called "CLK". You can set the association between block clock and top clock by using the **set_block_to_top_map** command, or by using **set_clock_trunk_endpoints -auto_clock** and specifying the same rule. See the **set_block_to_top_map** man page for more details. Note that the **-clocks** and **-auto_clock** options are mutually exclusive. You can select one of three rules for auto-clock mapping:

- **connected** - The "connected" rule associates block-level clocks with top-level clocks under the following conditions: If a block clock has its source at a block port (a boundary clock), it can be associated to a top-level clock that propagates to that port. If a block clock has its source on an internal pin sources (a local clock), the top-level clock source must be declared at the same pin.
- **local** - The "local" rule maps connected clocks. In addition, if a block clock has its source on an internal pin sources (a local clock) and has no associated top-level clock, a new top-level clock will be created with the same waveform as the block-level clock. The new top-level clock will be associated with the local block-level clock.
- **all** - The "all" rule maps clocks as in the "local" rule. In addition, if a block clock has its source at a block port (a boundary clock) and no top-level clock propagates to that port, a new top-level clock will be created with the same waveform as the block-level clock. The new top-level clock will be associated with the local block-level clock. The top-level boundary clock will allow timing analysis to be done, but it is not necessarily correct for signoff. You should consider removing or replacing this clock before signoff.

Note that if two block-level clocks are declared at the same boundary port, the "all" rule cannot resolve both of them. This is because it will not automatically declare two conflicting clocks in the top-level block. You must manually declare two top-level clocks, with the appropriate waveforms. After the top clocks exist, **-auto_clock** mapping can associate them with the block clocks.

DESCRIPTION

This command defines pins or ports as clock trunk endpoints for clock trunk planning. The clock trunk endpoints can be clock-specific. If you do not specify **-clock**, the tool applies the endpoint to all clocks.

In addition, you can specify a delay value from the endpoint to the clock sinks (flip-flops and other clocked cells) which should be assumed during clock trunk planing.

EXAMPLES

This example specifies block1/CLK as a clock trunk endpoint, and specifies a phase delay of 20 on the port.

```
prompt> set_clock_trunk_endpoints block1/CLK -delay 20
1
```

SEE ALSO

gui_load_clock_trunk_planning(2)
report_clock_trunk_endpoints(2)
remove_clock_trunk_endpoints(2)
synthesize_clock_trunks(2)

set_clock_uncertainty

Specifies the uncertainty (skew) of specified clock networks.

SYNTAX

string **set_clock_uncertainty**
uncertainty
[*object_list* |
-from *from_clock*
| -rise_from *rise_from_clock*
| -fall_from *fall_from_clock*
-to *to_clock*
| -rise_to *rise_to_clock*
| -fall_to *fall_to_clock*]
[-rise]
[-fall]
[-setup]
[-hold]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]

Data Types

uncertainty float
object_list list
from_clock list
rise_from_clock list
fall_from_clock list
to_clock list
rise_to_clock list
fall_to_clock list
mode_list list
corner_list list
scenario_list list

ARGUMENTS

-from *from_clock*

Specifies the source clock for interclock uncertainty. You must specify either the pair of **-from**/**-rise_from**/**-fall_from** and **-to**/**-rise_to**/**-fall_to**, or *object_list*; you cannot specify both.

-rise_from *rise_from_clock*

Same as the **-from** option, but indicates that *uncertainty* applies only to rising edge of the source clock. You can use only one of the **-from**, **-rise_from**, or **-fall_from** options.

-fall_from *fall_from_clock*

Same as the **-from** option, but indicates that *uncertainty* applies only to falling edge of the source clock. You can use only one of the **-from**, **-rise_from**, or **-fall_from** options.

-to *to_clock*

Specifies the destination clock for interclock uncertainty. You must specify either the pair of **-from**/**-rise_from**/**-fall_from** and **-to**/**-rise_to**/**-fall_to**, or *object_list*; you cannot specify both.

-rise_to *rise_to_clock*

Same as the **-to** option, but indicates that *uncertainty* applies only to rising edge of the destination clock. You can use only one of the **-to**, **-rise_to**, or **-fall_to** options.

-fall_to *fall_to_clock*

Same as the **-to** option, but indicates that *uncertainty* applies only to falling edge of the destination clock. You can use only one of the **-to**, **-rise_to**, or **-fall_to** options.

object_list

Specifies a list of clocks, ports, or pins for simple uncertainty. The uncertainty is applied either to capturing latches clocked by one of the clocks in *object list*, or capturing latches whose clock pins are in the fanout of a port or pin specified in *object_list*. You must specify either the pair of **-from** and **-to** options, or *object_list*; you cannot specify both.

-rise

Applies *uncertainty* to only the rising edge of the destination clock. By default, the uncertainty applies to both rising and falling edges. This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use **-rise_to** instead.

-fall

Applies *uncertainty* to only the falling edge of the destination clock. By default, the uncertainty applies to both rising and falling edges. This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use **-fall_to** instead.

-setup

Indicates that *uncertainty* applies only to setup checks. By default, *uncertainty* applies to both setup and hold checks.

-hold

Indicates that *uncertainty* applies only to hold checks. By default, *uncertainty* applies to both setup and hold checks.

-modes *mode_list*

Specifies the scenarios that the uncertainty value will be applied to. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option must not be specified together with **-modes**. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners *corner_list*

Specifies the scenarios that the uncertainty value will be applied to. If this option is given, all scenarios of the given corners and the current mode will be used. The **-scenarios** option must not be specified together with **-corners**. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios *scenario_list*

Specifies the scenarios that the uncertainty value will be applied to. The **-modes** or **-corners** option must not be specified together with **-scenarios**.

uncertainty

Specifies the uncertainty value. *uncertainty* is a floating point number. Typically, clock uncertainty should be positive. Negative uncertainty values are supported for constraining designs with complex clock relationships. Setting the uncertainty value to a negative number could lead to optimistic timing analysis and should be used with extreme care.

DESCRIPTION

Specifies the clock uncertainty (skew characteristics) of the specified clock networks. This command can specify either interclock uncertainty or simple uncertainty. For interclock uncertainty, use the **-from**/**-rise_from**/**-fall_from** and **-to**/**-rise_to**/**-fall_to** options to specify the source clock and the destination clock; all paths between these receive the uncertainty value. For simple uncertainty, use *object_list*; the uncertainty value is either to capturing latches clocked by one of the clocks in *object list*, or capturing latches whose clock pins are in the fanout of a port or pin specified in *object_list*.

Set the uncertainty to the worst skew expected to the endpoint or between the clock domains. You can increase the value to account for additional margin for setup and hold.

When you specify interclock uncertainty, ensure that you specify it for all possible interactions of clock domains. For example, if you specify paths from CLKA to CLKB and CLKB to CLKA you must specify the uncertainty for both, even if the values are the same. For an example, see the EXAMPLES section.

Interclock uncertainty is more specific than simple uncertainty. If a command that specifies interclock uncertainty conflicts with a command that specifies simple uncertainty, the command that specifies interclock uncertainty takes precedence. For an example, see the EXAMPLES section.

If there is no applicable interclock uncertainty for a path, the value for simple uncertainty is used. For an example, see the EXAMPLES section.

Clock uncertainty is managed on a per-scenario basis. For designs with multiple scenarios, you can specify different uncertainties for different scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the given uncertainty is applied to the current scenario. The command issues an error if there is no current scenario. If the **-scenarios** option is given, the uncertainty will be applied to all of the specified scenarios. If the **-modes** option is given, the uncertainty will be applied to all scenarios of the specified modes. The command issues an error if none of the specified modes have any scenarios. If the **-corners** option is given, the uncertainty will be applied to all scenarios of the specified corners and the current mode. The command issues an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are given, the uncertainty will be applied to all scenarios of the specified corners and the specified modes. The command issues an error if there are no such scenarios. It is an error to specify **-scenarios** with **-modes** or **-corners**.

To remove the uncertainties set by **set_clock_uncertainty**, use the **remove_clock_uncertainty** command.

To view clock uncertainty information, use the **report_clocks -skew** command.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example specifies that all paths leading to registers or ports clocked by CLK have setup uncertainty of 0.65 and hold uncertainty of 0.45.

```
prompt> set_clock_uncertainty -setup 0.65 [get_clocks CLK]
prompt> set_clock_uncertainty -hold 0.45 [get_clocks CLK]
```

The following example specifies interclock uncertainties between PHI1 and PHI2 clock domains.

```
prompt> set_clock_uncertainty 0.4 -from PHI1 -to PHI1
prompt> set_clock_uncertainty 0.4 -from PHI2 -to PHI2
prompt> set_clock_uncertainty 1.1 -from PHI1 -to PHI2
prompt> set_clock_uncertainty 1.1 -from PHI2 -to PHI1
```

The following example specifies interclock uncertainties between PHI1 and PHI2 clock domains with specific edges.

```
prompt> set_clock_uncertainty 0.4 -rise_from PHI1 -to PHI2
prompt> set_clock_uncertainty 0.4 -fall_from PHI2 -rise_to PHI2
prompt> set_clock_uncertainty 1.1 -from PHI1 -fall_to PHI2
```

The following example shows conflicting **set_clock_uncertainty** commands, one for simple uncertainty and one for interclock uncertainty. The interclock uncertainty value of 2 takes precedence.

```
prompt> set_clock_uncertainty 5 [get_clocks CLKA]
prompt> set_clock_uncertainty 2 -from [get_clocks CLKB] -to [get_clocks CLKA]
```

The following example specifies the uncertainty from CLKA to CLKB and from CLKB to CLKA. Notice that both must be specified even though the value is the same for both.

```
prompt> set_clock_uncertainty 2 -from [get_clocks CLKA] -to [get_clocks CLKB]
prompt> set_clock_uncertainty 2 -from [get_clocks CLKB] -to [get_clocks CLKA]
```

The following example illustrates a situation in which simple uncertainty is used when there is no applicable interclock uncertainty for a path. The first command specifies a simple uncertainty of 5 for CLKA paths, and the second command specifies an interclock uncertainty of 2 for paths from CLKB to CLKA. If there are paths between CLKA and other clocks (for example, CLKC, CLKD, ...) for which interclock uncertainty has not been specifically defined, the simple uncertainty (in this case, 5) is used.

```
prompt> set_clock_uncertainty 5 [get_clocks CLKA]
prompt> set_clock_uncertainty 2 -from [get_clocks CLKB] -to [get_clocks CLKA]
```

SEE ALSO

remove_clock_uncertainty(2)
report_clocks(2)
set_clock_latency(2)
set_clock_transition(2)

set_colors

Sets a user-specified color on objects and their child objects.

SYNTAX

int **set_colors**
[-color *color_id*]
[-color_name *color_name*]
[-cycle_color]
[-hierarchy_types all | normal | boundary | soft_macro]
[-depth *hierarchy_depth*]
[-keep]
[*object_list*]

Data Types

color_id integer
color_name string
hierarchy_depth integer
object_list collection

ARGUMENTS

-color *color_id*

Specifies a color ID, which is used to set a color on objects. Valid *color_id* values are integers from 1 to 64.

-color_name *color_name*

Specifies a color name, which is used to set a color on objects. Valid values are: purple, blue, green, orange, red, light_purple, light_blue, light_green, light_orange, and light_red.

-cycle_color

Generates automatically different colors for each given hierarchical cell and the cells descended from the hierarchical cell, or for each given voltage area and the cells associated with the voltage area. If **-color** is also given, the color starts from the given ID. If **-color** is not given, the color starts with the ID following the last used color ID during auto-cycling.

A color ID is used to color all leaf cells in the given collection of cells, or all leaf cells owned by the top design if no collection is given. The ID is then incremented and is used to color the first hierarchical cell in the collection and its descendents, or the first voltage area and the associated cells. The ID is then incremented again and is used to color the second hierarchical cell in the collection and its descendents, or the second voltage area and its cells, and so on.

-hierarchy_types all | normal | boundary | soft_macro

Colors the hierarchical cells of the specified type. The command traverses the design hierarchy starting from the given cells to the

depth given by the **-depth** option argument and colors hierarchical cell of the given types with different colors. The allowed types are *normal*, *boundary*, *soft_macro*, or *all*. The last value, *all*, matches any hierarchical cell type.

If no cell collection argument is given, traversal of the hierarchy begins from the top design. If a cell collection is given, all hierarchical cells of given type in the given collection are colored with cycled colors.

If no **-depth** option argument is given but a cell collection is given, then the default depth of *0* is used, coloring hierarchical cells in the given cell collection. If neither a **-depth** option argument or a cell collection are given, then the default depth of *1* is used, coloring hierarchical cells that are the immediate children of the top design.

The **-hierarchy_types** option implies **-cycle_color**.

-depth *hierarchical_depth*

Specifies the levels of hierarchy to traverse to find hierarchical cells to color. This option is only meaningful when given with the **-hierarchy_types** option. The command traverses the design hierarchy starting from the given cells to the *hierarhcical_depth* value given to find cells of matching type to color.

A depth value of *0* traverses only the given cells. A depth value of *1* traverses the immediate children of the given cells. The special depth value of *all* means to traverse the hierarchy all the way to leaf descendents to find cells of the given type to color.

-keep

This option specifies that if a cell already has a color attribute, leave the current color undisturbed. This option is useful when incrementally adding object coloring.

object_list

Specifies a collection of objects on which to set the colors. The collection can include cells, designs, and voltage areas. If omitted, colors are set on cells starting from the current top design.

DESCRIPTION

The **set_colors** command sets the colors on cells descended from top hierarchical cells in the current design, or hierarchical cells in the specified collection, or voltage areas and their cells. The colors set on objects are used to paint the objects in some graphic views, such as the Layout View or the Hierarchy View.

If *objects* are specified and contains cells, the command sets the color on all cells in the collection and all cells descended from hierarchical cells in the collection. If *objects* are specified and contains voltage areas, the command sets the color of all the voltages areas in the collection and the cells associated with the voltage areas. If *objects* are not specified, the command sets the color of all cells in the current design.

When **-color** is specified, the specified color is used. If it is not specified, then the current default color is used. The set_colors command uses colors from the highcontrast color palette.

When **-cycle_color** is given and *objects* contains cells, the command generates a different color for each hierarchical cell in the collection, and sets the color on all cells descended from that hierarchical cell. All non-hierarchical cells in the collection are colored with one color different from the hierarchical cell colors.

When **-cycle_color** is given and *objects* contains voltage areas, the command generates a different color for each voltage area in the collection, and sets the voltage area color on all cells associated with the voltage area.

When **-cycle_color** is specified without *objects*, the command generates a different color for each top-level hierarchical cell in the current design, and sets the color on all cells descended from these hierarchical cells.

Hierarchy traversal and cell coloring stops at an instance of a referenced design, such as a soft macro. The instance itself is colored, but its children will not be colored.

EXAMPLES

The following example generates different colors for each top hierarchical cell in the current design, and sets the color on all descended cells in the hierarchical cell:

```
prompt> set_colors -cycle_color
```

The following example sets different colors on all physical boundaries, including hierarchy boundaries and committed soft macros, in the top design:

```
prompt> set_colors -depth all \
    -hierarchy_types {boundary soft_macro} [get_designs]
```

The following example sets different colors on all normal hierarchies in the top design. Note that the top design is implied by omitting the cells on which to operate:

```
prompt> set_colors -depth all \
    -hierarchy_types normal
```

The following example sets the color on all immediate hierarchical child cells of I_ORCA_TOP:

```
prompt> set_colors -hierarchy_types all \
     -depth 1 [get_cells I_ORCA_TOP]
```

The following example sets the color with color ID 30 on I_ORCA_TOP and all of its descendant cells while leaving already colored cells untouched:

```
prompt> set_colors -color 30 \
     -depth all [get_cells I_ORCA_TOP]
```

The following example sets the color with color Id 8 on RES_I113 cell and all cells inside the RES_I113 hierarchical cell:

```
prompt> set_colors -color 8 [get_cells RES_I113]
```

The following example sets the color with color Id 7 on the voltage area named myVoltageArea, and all cells associated with the voltage area.

```
prompt> set_colors -color 7 [get_voltage_areas myVoltageArea]
```

SEE ALSO

remove_colors(2)
gui_get_color_value(2)

set_command_option_value

Set a command option default or current value.

SYNTAX

string **set_command_option_value**
-default | -current

-command *command_name*

-option *option_name* | -position *positional_option_position*

-value *value* | -undefined

ARGUMENTS

-default

Set the default option value.

-current

Set the current option value.

-command *command_name*

Set the option value for an option of this command.

-option *option_name*

Set the option value for this option of the command.

-position *positional_option_position*

Set the option value for this positional option of the command.

-value *value*

Set the option value to this value.

-undefined

Set the option value to the "undefined value".

DESCRIPTION

Sets the current or default value of a command option.

Either -default must be specified (to specify setting the default value of the option), or -current must be specified (to specify setting the current value of the option). Unlike for get_command_option_values, one of -current or -default must be specified for set_command_option_value.

An option of a command must be specified (for value setting) by passing a command name via -command and either an option name (for an option of the command) via -option, or the position of a positional option via -position. The *option_name* passed via -option (for a dash option) must not have its leading dash character omitted. An option name passed via -option may be either the name of a dash option or the name of a positional option. A positional option may be specified either via -option or via -position. Positional option positions of a command are numbered 0, 1, 2, ... (N-1), where N is the number of positional options of the command.

To set the (current or default) value of the option, a string value should be passed via -value. Alternatively, you can pass -undefined to reset the option to have the "undefined value". (CAVEAT: For some "set value implementations" for numeric options, -undefined does not really "undefine" the value - instead the value is set to 0.)

Processing of the -value value does not include any CCI option checking of the value (such as checking whether the option value has correct syntax for its type).

The command "throws" a Tcl error for various conditions:
* the command does not exist
* the command exists but no such option of the command exists
* the set operation failed (could be due to a failed conversion for one of the "set value" implementations which does a conversion from string to one of integer, double, etc.)

The -undefined option is a mutually exclusive alternative to the -value option. -undefined has the effect of resetting the option value to the "undefined value".

A possible power user application: power users could put set_command_option_value commands in their home directory Tcl startup files for their favorite dialogs/commands to "seed" initial current values for these dialogs. This may lower the need for automatically saving replay scripts (of set_command_option_value commands for each command/dialog option/field) on exit from the application.

EXAMPLES

The following example sets the current value for the -bar option of the foo command to a value of abc.

```
prompt> set_command_option_value -current -command foo \
  -option "-bar" -value abc
```

The following example sets the current value for the first positional argument of the foo command to a value of xyz.

```
prompt> set_command_option_value -current -command foo \
  -position 1 -value xyz
```

SEE ALSO

get_command_option_values(2)
preview(2)

set_consistency_settings_options

Sets the corresponding options for the **check_consistency_settings** command. We need to specify some executable image and related script file before we run command **check_consistency_settings**.

SYNTAX

status **set_consistency_settings_options**
[-exec_path *path*]
[-tool *type*]
[-script *filename*]
[-saved_session *directory*]
[-tluplus *tluplus_filename*]
[-corner *corner_name*]
[-star_corner *corner_name*]
[-early]

Data Types

path string
filename string
directory string
type string
corner_name string
corner_name string

ARGUMENTS

-exec_path *path*

Specifies the UNIX path to the shell executable, like pt_shell. The path should be absolute.

-tool

Indicates the executable image type. The valid values are PT, ICC, ICC2, StarRC, and PrimePower.

-script *filename*

Specifies a Tcl file or command file that is sourced by the executable tool immediately after the tool is invoked.

-saved_session *directory*

Specifies a directory which was generated by PrimeTime command 'save_session' previously.

Tool restores PrimeTime design status based upon this directory. Then, collect corresponding variable settings for correlation check between the implementation tool and PrimeTime.

Option '-script' will be override by this option if specified.

-tluplus *tluplus_filename*

Specifies a process technology file (TLUPlus) that is using for process information checking with the StarRC process technology file (nxtgrd). If not specifies TLUPlus file, then will skip process information checking. This feature only supports StarRC version 2015.12-SP2 or later and encrypted TLUPlus file version 16 or later, otherwise process information checking will skip with Error messages. Every versions of non-encrypted TLUPlus files are supported.

-corner

Specifies a corner name. The values could be any pre-defined system corner name. This option may only use for extraction consistency settings. Default value is current corner of current scenario.

-star_corner

Specifies a StarRC corner name. The values could be any pre-defined StarRC corner name in its SMC(Simulteneous Multi Corner) scripts. This option may only use for extraction consistency settings in case of StarRC corner name is different than the corner name in the implementation tool. Default value is same corner name as the default corner name or -corner <corner_name> if -corner option has used.

-early

Specifies a corner type (early/late). This option may only use for extraction consistency settings. Default value is type late.

DESCRIPTION

This command sets the executable location and related running scripts for **check_consistency_settings**. With these information, correlation checker will automatically collect the variables settings information and then do consistency check.

Multicorner-Multimode Support

This command uses information for current scenario. Or user may specify specific corner name and type (early/late) for extraction consistency settings.

EXAMPLES

The following example runs the **set_consistency_settings_options** command.

```
prompt> set_consistency_settings_options \
  -exec_path /PrimeTime/linux/syn/bin -tool pt -script example.tcl
prompt> set_consistency_settings_options \
  -exec_path /StarRC/linux/syn/bin -tool starrc -script example.tcl \
  -corner maxCorner
```

SEE ALSO

check_consistency_settings(2)

set_constant_register_removal

Sets the **remove_constant_register** attribute on the specified cells or modules, allowing removing of the constant registers.

SYNTAX

status **set_constant_register_removal**
obj_list
[true | false]

Data Types

obj_list list

ARGUMENTS

obj_list

Specifies a list of cell or module names for which to remove constant registers. Cell names in *obj_list* must be from the current design. If multiple objects are specified, they must be enclosed in quotation marks ("") or braces ({}).

true | **false**

Specifies the value with which to set the **remove_constant_register** attribute. When this option is set to true (the default), constant register removal is enabled on the specified objects (cells or modules). Constant register removal is performed on the specified objects provided that the application option **compile.seqmap.remove_constant_registers** is set to true.

A setting of false on a module do not remove constant registers for the whole module. A setting of false on an instance do not remove constant registers for the whole instance.

DESCRIPTION

The **set_constant_register_removal** command sets the **remove_constant_register** attribute on the specified *obj_list*. This attribute is used to specify that the cells or modules are to be remove constant registers.

If a cell with a specified name is found in the current design, the **remove_constant_register** attribute is set to the specified value on the cell. If no cell with a specified name is found, the tool searches for a module and the attribute set on the module.

EXAMPLES

The following example shows how to enable register removal during optimization for the sequential cells named U0 and U1 and how to disable register removal for the U2 cell:

```
prompt> set_constant_register_removal {U0 U1} true
prompt> set_constant_register_removal U2 false
```

SEE ALSO

compile(2)
get_attribute(2)

set_constraint_mapping_file

Loads a map file that contains a list of blocks and corresponding UPF, SDC, DEF, timing budget, blackbox timing, CTS constraint or PG constraint files for the blocks.

SYNTAX

status **set_constraint_mapping_file**
[-reset]
filename

Data Types

filename string

ARGUMENTS

-reset

Resets the constraint map for the current design.

This option can be used independently to remove the map from the current design. It can also be used together with the *fileName* option to replace the map saved with the current design with the new map in the provided *fileName*.

filename

Specifies the name of the file that contains block constraint map information.

When specified together with the *-reset* option, the map in the specified file replaces the existing map saved with the current design. When this option is specified without the *-reset* option, the map in the specified file is appended to the existing map saved with the current design.

DESCRIPTION

This command reads a file that contains a list of block names, constraint types, and constraint file names, and assigns the constraint file to the specified block. This command is used to assign UPF, SDC, DEF, timing budget, blackbox timing, clock nets, CTS constraint, PG constraint or floorplan files to specific blocks.

The syntax for each line in the mapping file is:

block_design_name constraint_type file_name

block_design_name is the reference block name for either the top-level design or a block-level module.

constraint_type is one of the following keywords:

- BUDGET
- CLKNET
- COMPILE_PG
- DEF
- ETM_UPF
- PG_CONSTRAINT
- CTS_CONSTRAINT
- SDC
- UPF
- BTM
- FLOORPLAN
- SCANDEF

file_name is the constraint file name with its UNIX path.

The **BUDGET** constraint type specifies the constraint file that contains timing budget information for the corresponding block.

The **CLKNET** constraint type specifies the file that contains attribute setting commands that would tell the tool which nets are clock nets. The file is typically generated by split_constraints.

The **COMPILE_PG** constraint type specifies the constraint file that contains application option settings and setup commands for the **compile_pg** command.

The **DEF** constraint type specifies that the file is a DEF file for the corresponding block.

The **ETM_UPF** constraint type specifies that the constraint file contains UPF constraints for the corresponding ETM block.

The **PG_CONSTRAINT** constraint type specifies that the constraint file contains power planning constraints for the corresponding block.

The **SDC** constraint type specifies that the constraint file contains timing constraints for the corresponding block, including mode and corner setups.

The **UPF** constraint type specifies that the constraint file contains UPF constraints for the corresponding block.

The **BTM** constraint type specifies that the constraint file contains blackbox timing model constraints for the corresponding block.

The **FLOORPLAN** constraint type specifies that the constraint file contains floorplan information of the corresponding block. The file is typically generated by write_floorplan.

The **SCANDEF** constraint type specifies that the file is a scan DEF file for the corresponding block.

Depending on the flow you are using, the **shape_blocks**, **create_placement -floorplan**, **estimate_timing**, **create_abstract -all_blocks**, and **create_abstract -cells {cell_list}** commands might require design or abstract views. To create the views, certain constraints must be loaded so the abstract view can access the constraint information. To make the constraints available, the **set_constraint_mapping_file** command sets the association between the blocks and the corresponding constraint files for the blocks.

During abstract creation with the **shape_blocks**, **create_placement -floorplan**, **estimate_timing**, **create_abstract -all_blocks**, or **create_abstract -cells** commands, the UPF constraint file is loaded internally with the **load_upf** command, if no UPF was loaded for

the block.

The SDC constraint file is loaded internally with the **source** command, if the block has no SDC constraint. The timing budget constraint file is always loaded internally with the **source** command.

The CLKNET constraint file is loaded internally with the **source** command. Once it's loaded, the block would have attribute *is_clknet_file_loaded* set to true, and the CLKNET constraint file would not be loaded again.

The DEF and scan DEF constraint file is loaded with the **read_def** command only when the tool detects that the block has no design view; the tool automatically expands the outline view to generate the design view.

The CTS constraints are loaded with **source** command when creating the block's estimated abstract.

If a block is blackbox and has BTM constraint, its BTM constraint would be loaded automatically with **source** command after the tool loads the SDC constraint for the block.

The user can also manually ask to the tool to load specific type of constraint for specific blocks with command **load_block_constraints**.

The FLOORPLAN constraint file is only loaded when the user specifically asks the tool to load it for specific blocks with command **load_block_constraints**.

EXAMPLES

The following example specifies the DEF, UPF and FLOORPLAN constraint files. The files are used in the hierarchical flow from Verilog input to block shaping. The design has a top-level and two child blocks: BLK_INST_A (whose reference design is BLK_A) and BLK_INST_B (whose reference design is BLK_B). The content of the constraint mapping file *map_file.1* is:

```
BLK_A  DEF ./constraints/BLK_A.def
BLK_A  UPF ./constraints/BLK_A.upf
BLK_A  FLOORPLAN ./floorplan/BLK_A/BLK_A.tcl
BLK_B  DEF ./constraints/BLK_B.def
BLK_B  UPF ./constraints/BLK_B.upf
BLK_B  FLOORPLAN ./floorplan/BLK_B/BLK_B.tcl
TOP    DEF ./constraints/TOP.def
TOP    UPF ./constraints/TOP.upf
TOP    FLOORPLAN ./floorplan/TOP.tcl
```

The following Tcl commands are used in the flow. The **shape_blocks** command determines that the two blocks are both outline views. The command loads each block, expands the outline view to generate design view, loads the corresponding DEF constraint file, loads the UPF constraint file, then creates the placement abstract view for the block. After generating placement abstract views for both blocks, the command loads the top-level design, expands the top-level outline view, loads the DEF for the top-level design, and loads the UPF for the top-level design. After these steps, block shaping is performed.

```
prompt> read_verilog_outline -top TOP design.v
prompt> commit_block BLK_A
prompt> commit_block BLK_B
prompt> set_constraint_mapping_file map_file.1
prompt> shape_blocks
```

The following Tcl commands can be used to load floorplan to the hierarchical design. When **load_block_constraints** is issued, the tool would see that the top level and the child blocks are outline view, then it would expand the child blocks to design view, load their corresponding floorplan files, create a placement abstract view for them, and come to the top level, expand it to design view and change the child blocks to reference to their abstract views, then load the top level floorplan file.

```
prompt> read_verilog_outline -top TOP design.v
```

```
prompt> commit_block BLK_A
prompt> commit_block BLK_B
prompt> set_constraint_mapping_file map_file.1
prompt> load_block_constraints -all_blocks -type FLOORPLAN
```

The following example specifies a constraint mapping file and runs **estimate_timing** on the design. The tool opens each block, loads the respective SDC constraint file, then loads the corresponding BUDGET constraint file, and creates a timing abstract with estimate_timing data, loads the top-level design, and loads the top-level SDC constraint file. In the final step, estimate timing at the top-level is performed. The content of the constraint mapping file *map_file.2* is:

```
BLK_A  SDC    ./constraints/BLK_A.sdc.tcl
BLK_B  SDC    ./constraints/BLK_B.sdc.tcl
BLK_A  BUDGET ./budgets/BLK_A.budget.tcl
BLK_B  BUDGET ./budgets/BLK_B.budget.tcl
TOP    SDC    ./constraints/TOP.sdc.tcl
```

The commands to perform the estimate_timing step by using the *map_file.2* are:

```
prompt> set_constraint_mapping_file map_file.2
prompt> estimate_timing
```

SEE ALSO

report_constraint_mapping_file(2)
load_block_constraints(2)
create_abstract(2)
create_placement(2)
estimate_timing(2)
merge_abstract(2)
shape_blocks(2)

set_corner_status

Configures analysis settings for mode and corner combinations (obsolete).

SYNTAX

status **set_corner_status**
[-modes *mode_list*]
[-corners *corner_list*]
[-setup true | false]
[-hold true | false]
[-power true | false]
[-max_transition true | false]
[-max_capacitance true | false]
[-min_capacitance true | false]
[-active true | false]
[-label *label*]

Data Types

mode_list list
corner_list list
label string

ARGUMENTS

-modes *mode_list*

Specifies the mode names to configure. The command configures each combination of the modes specified the **-modes** option and the corners specified by the **-corners** option, based on other command options. If this option is not specified, the command uses the current mode. If neither a mode nor a corner list is given, the current-mode/current-corner combination is configured.

-corners *corner_list*

Specifies the corner names to configure. The command configures each combination of the modes specified the **-modes** option and the corners specified by the **-corners** option, based on other command options. If this option is not specified, the command uses the current corner. If neither a mode nor a corner list is given, the current-mode/current-corner combination is configured.

-setup true | false

Specifies whether setup analysis is activated for the specified mode and corner combinations.

-hold true | false

Specifies whether hold analysis is activated for the specified mode and corner combinations.

-power true | false

Specifies whether power analysis is activated for the specified mode and corner combinations.

-max_transition true | **false**

Specifies whether max-transition DRC is activated for the specified mode and corner combinations.

-max_capacitance true | **false**

Specifies whether max-capacitance DRC is activated for the specified mode and corner combinations.

-min_capacitance true | **false**

Specifies whether min-capacitance DRC is activated for the specified mode and corner combinations.

-active true | **false**

Specifies whether any previous or current analysis settings (setup, hold, power, etc) are active or disabled. If false, all analysis settings are temporarily disabled and can be reactivated by specifying **-active true** at a later time.

-label ***label***

Assigns the specified *label* to the mode and corner combinations. The label does not affect analysis and is only printed in reports created by the **report_timing** and **report_qor** commands. Multiple mode and corner combinations can have the same label. A blank string is a legal label value.

DESCRIPTION

This command enables analysis and DRC checking for mode and corner combinations in the current design. Setup analysis, hold analysis, and various types of DRC checking can be configured independently. By default, all analysis or DRC checking is done for a mode and corner combination. This command can be used to setup the analysis and DRC checking for only the appropriate mode and corner combinations.

Note: This command is obsolete, and has been replaced by the **set_scenario_status** command. Use **set_scenario_status** instead.

Multicorner-Multimode Support

SEE ALSO

create_scenario(2)
set_scenario_status(2)
report_scenario(2)

set_current_command_mode

SYNTAX

string **set_current_command_mode**

-mode *command_mode* | -command *command*

string *command_mode*
string *command*

ARGUMENTS

-mode *command_mode*

Specifies the name of the new command mode to be made current. **-mode** and **-command** are mutually exclusive; you must specify one, but not both.

-command *command*

Specifies the name of the command whose associated command mode is to be made current. **-mode** and **-command** are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **set_current_command_mode** sets the current command mode. If there is a current mode in effect, the current mode is first canceled, causing the associated clean up to be executed. The initialization for the new command mode is then executed as it is made current.

If the name of the given command mode is empty, the current command mode is cancelled and no new command mode is made current.

A current command mode stays in effect until it is displaced by a new command mode or until it is cleared by an empty command mode.

If the command completes successfully, the new current command mode name is returned. On failure, the command returns the previous command mode name which will remain current and prints an error message unless error messages are suppressed.

EXAMPLES

The following example sets the current command mode to a mode called "mode_1"

shell> **set_current_command_mode -mode mode_1**

The following example clears the current command mode without setting a new mode.

shell> **set_current_command_mode -mode ""**

The following example sets the current command mode to the mode associated with the command "modal_command"

shell> **set_current_command_mode -command modal_command**

set_current_ems_database

Sets the current EMS database.

SYNTAX

collection **set_current_ems_database**
ems_database
[-reset]

Data Types

ems_database string

ARGUMENTS

ems_database

Sets *ems_database* as the current EMS database. The database must be opened and in memory. Use **get_ems_databases** to view the list of currently opened databases.

-reset

Removes the current EMS database. The database still remains opened and in memory.

DESCRIPTION

The **set_current_ems_database** command sets the specified EMS database as the current EMS database. The specified EMS database must be opened with the **open_ems_database** command or created with the **create_ems_database** or **check_design -open_message_browser** command, otherwise **set_current_ems_database** issues an error message. The **-reset** option clears the current EMS database, but leaves the database in memory.

EXAMPLES

The following example sets the current EMS database to EMS_DB_0.ems.

```
prompt> set_current_ems_database EMS_DB_0.ems
prompt> get_current_ems_database
{EMS_DB_0.ems}
```

The following example clears the current EMS database.

```
prompt> get_current_ems_database
{EMS_DB_0.ems}
prompt> set_current_ems_database -reset
prompt> get_current_ems_database
```

SEE ALSO

close_ems_databases(2)
create_ems_database(2)
get_ems_databases(2)
open_ems_database(2)
save_ems_database(2)
get_current_ems_database(2)

set_current_mismatch_config

Sets the current mismatch config

SYNTAX

status **set_current_mismatch_config**
config_name
[-enable library | netlist | upf | routing]

Data Types

config_name string

ARGUMENTS

config_name

Specify the mismatch config name to be set as current.

-enable

Specify list of categories to be enabled. All the mismatch types under that category will be enabled.

DESCRIPTION

The set_current_mismatch_config command sets the current mismatch config. The named mismatch config should exist, else, the command returns error. Any pre-defined or user defined configs can be used to set as current. -enable is optional. Without that, only the current mismatch config will be changed. If -enable is given, it will change the current mismatch config, and additionally, it will enable the mismatches of the specified category for the newly set config. The Action and Repair strategy of the enabled mismatches will match those of the pre-defined "auto_fix" config. In order to query the list of mismatch types corresponding to a category, the "category" attribute on the mismatch_type object can be queried.

EXAMPLES

The following example sets the current mismatch config to user_def1 and enables repairing of netlist and library related mismatches:

prompt> **set_current_mismatch_config user_def1 -enable {netlist library}**

```
1
prompt> get_current_mismatch_config
user_def1
prompt> set_current_mismatch_config user_def2
Warning: It is not recommended to switch between configs because there might
be some design mismatches already repaired in previous config. And those
mismatches will not be in sync with current config anymore. (DMM-010)
1
prompt> get_current_mismatch_config
user_def2
```

The following example uses non-existing mismatch config:

```
prompt> set_current_mismatch_config dont_exist
Error: Mismatch config 'dont_exist' do not exists. (DMM-015)
0
```

SEE ALSO

report_design_mismatch(2)
get_mismatch_types(2)
get_mismatch_objects(2)
get_current_mismatch_config(2)
report_mismatch_configs(2)
create_mismatch_config(2)

set_data_check

Sets data-to-data checks using the specified values of setup and hold time.

SYNTAX

status **set_data_check**
  -modes *mode_list*
  -corners *corner_list*
  -scenarios *scenario_list*
  -from *from_object*
    | -rise_from *from_object*
    | -fall_from *from_object*
  -to *to_object*
    | -rise_to *to_object*
    | -fall_to *to_object*
  [-setup | -hold]
  [-clock *clock_object*]
  [*check_value*]

Data Types

from_object	collection of 1 object
to_object	collection of 1 object
clock_object	collection of 1 object
check_value	float

ARGUMENTS

-modes *mode_list*

Set data check for scenarios of these modes (default is current mode, or current scenario if neither mode nor corner is specified)

-corners *corner_list*

Set data check for scenarios of these corners (default is current scenario if neither mode nor corner is specified)

-scenarios *scenario_list*

Set data check for these scenarios (default is current scenario)

-from *from_object*

Specifies a pin or port in the current design as the related pin of the data-to-data check to be set. Both rising and falling delays are checked.

You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-to ***to_object***

Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be set. Both rising and falling delays are constrained.

You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-rise_from ***from_object***

Specifies a pin or port in the current design as the related pin of the data-to-data check to be set. This option applies only to rising delays at the related pin.

You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-fall_from ***from_object***

Specifies a pin or port in the current design as the related pin of the data-to-data check to be set. This option applies only to falling delays at the related pin.

You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-rise_to ***to_object***

Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be set. This option applies only to rising delays at the constrained pin.

You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-fall_to ***to_object***

Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be set. This option applies only to falling delays at the constrained pin.

You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-setup

Indicates that the data check value is for setup analysis only. If neither **-setup** nor **-hold** is specified, the value applies to both setup and hold.

-hold

Indicates that the data check value is for hold analysis only. If neither **-setup** nor **-hold** is specified, the value applies to both setup and hold.

-clock ***clock_object***

Specifies a single clock that launches the signal for the related pin of the data check.

check_value

Specifies the value of the setup or hold time for the check.

DESCRIPTION

The **set_data_check** command specifies a data-to-data check to be performed between the from object and the to object using the

specified setup or hold value.

The pulse relation between clocks of the related pin and the constrained pin is considered to be zero cycles. The normal sequential check uses one cycle. Data checks on clock pins are not supported. Both constraint point and related point of the check have to be on the data network for a valid check.

The data check is treated as nonsequential; that is, the path goes through the related and constrained pins and is not broken at these pins. To report the constraint related to the data check, use the **report_timing** command with the **-to** option to specify the data-check-constrained pin.

To remove information set by the **set_data_check** command, use the **remove_data_check** command.

Multicorner-Multimode Support This command applies to the current scenario only.

EXAMPLES

The following example creates a data-to-data check from pin and1/B to pin and1/A with respect to the rising edge of the signal at and1/B, using a setup and hold time of 0.4.

```
prompt> set_data_check -rise_from and1/B -to and1/A 0.4
```

The following example creates a data-to-data check from pin and1/B to pin and1/A with respect to the rising edge of the signal at and1/B, coming from the clock domain of CK1, and constraining only the falling edge of the signal at and1/A, using a setup time of 0.5 and no hold check.

```
prompt> set_data_check -rise_from and1/B -fall_to and1/A \
  -setup -clock [get_clock CK1] 0.5
```

SEE ALSO

remove_data_check(2)
report_timing(2)
update_timing(2)

set_datapath_architecture_options

Controls the strategies used when generating the datapath cell for arithmetic and shift operators.

SYNTAX

status **set_datapath_architecture_options**
[-all_options auto | true | false]
[-booth_encoding auto | true | false]
[-booth_radix8 auto | true | false]
[-booth_mux_based auto | true | false]
[-booth_cell auto | true | false]
[-mult_radix4 auto | true | false]
[-mult_nand_based auto | true | false]
[-mult_sign_mag auto | true | false]
[-tp_opt_tree auto | true | false]
[-tp_oper_sel auto | true | false]
[-inv_out_adder_cell auto | true | false]
[-4to2_compressor_cell auto | true | false]
[-optimize_for default | area | speed | area,speed]
[-mult_arch *architecture*]
[*module_or_cell_list*]

ARGUMENTS

-all_options auto | true | false

Specifies the value for all datapath architecture options. When **true** or **false** is specified, **-optimize_for** is not affected.

The **set_datapath_architecture_options** command always processes the value of the **-all_options** option before any other options when they are used together in the same command line.

-booth_encoding auto | true | false

Controls the uage of Booth-encoding architectures to implement multipliers.

When set to **auto**, the default, the tool uses the option only if there is a QoR benefit. When you set the **-booth_encoding** option to **true**, the tool always uses the option. When you set the **-booth_encoding** option to **false**, the tool never uses the option.

-booth_radix8 auto | true | false

Controls the usage of radix-8 Booth-encoding architectures to implement multipliers. This option is active only if the **-booth_encoding** option is set to **auto** or **true**.

When you set the **-booth_radix8** option to **auto**, the default, the tool uses the **-booth_radix8** option option only if there is a QoR benefit. The tool always uses the **-booth_radix8** option when it is set to **true** and never uses the **-booth_radix8** option when it is set to **false**.

-booth_mux_based auto | true | false

Controls the usage of MUX-based Booth encoding. MUX-based encoding provides better QoR for technologies with fast multiplexer cells. Otherwise, the tool uses XOP-based Booth encoding.

When you set the **-booth_mux_based** option to **auto**, the default, the tool uses the option only used if it determines there is a QoR benefit. When you set the **-booth_mux_based** option to **true**, the tool always uses the option. When you set the **-booth_mux_based** option to **false**, the tool never uses the option.

-booth_cell auto | true | false

Controls the usage of special Booth multiplier cells in the target library for compilation.

When you set the **-booth_cell** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-booth_cell** option to **true**, the tool always uses the option if applicable. When you set the **-booth_cell** option to **false**, the tool never uses the option.

This option may not work if the special Booth multiplier cells are not availabe in the library.

-mult_radix4 auto | true | false

Controls the use of radix-4 non-Booth architectures to implement multipliers. This option cannot be used if the **-booth_encoding** option is set to **true**. The **-booth_encoding** option must be inactive (set to **auto** or **false**).

When you set the **-mult_radix4** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-mult_radix4** option to **true**, the tool always uses the option. When you set the **-mult_radix4** option to **false**, the tool never uses the option. This option is only available in when total power mode is on.

-mult_nand_based auto | true | false

Controls the usage of NAND-based multiplier structures. Non-Booth multipliers use AND gates or an INVERT-NOR structure to generate the partial-product bits. This option allows you to use the NAND gates to produce inverted bits that you can add directly. This can result in lower dynamic power and lesser glitches. This option is active only if the **-booth_encoding** option is set to **auto** or **false**.

When you set the **-mult_nand_based** option to **auto**, the default, the tool uses the option only when the total power mode is enabled and if it determines there is a QoR benefit. When you set the **-mult_nand_based** option to **true**, the tool always uses the option. When you set the **-mult_nand_based** option to **false**, the tool never uses the option.

-mult_sign_mag auto | true | false

Controls the usage of sign-magnitude multiplier architecture. When low-magnitude inputs into a signed multiplier change the sign frequently, all sign-extension bits toggle and introduce a lot of activity into the multiplier circuit. The sign-magnitude technique converts 2's-complement representation of inputs into sign-magnitude representation, and the multiplication is carried out on the unsigned magnitude alone without any sign-extension bits. The output is converted back to 2's-complement. This results in lower activity and dynamic power for signed multipliers with low magnitude inputs.

When you set the **-mult_sign_mag** option to **false**, the default, the tool never uses the option. When you set the **-mult_sign_mag** option to **true**, the tool always uses the option. Setting the **-mult_sign_mag** option to **auto** has the same effect as **false** since automatic selection on it is currently not supported.

This architecture usually degrades QoR of signed multipliers because it increases delay, area and dynamic power as reported by the tool. Dynamic power is only reduced when inputs have low magnitude most of the time (i.e. upper bits are just sign-extension and therefore correlated). Power reduction can only be measured using back-annotated activity from netlist simulation, because the synthesis tool does not process information about correlated input bits. Therefore, this option should only be set for individual signed multipliers that are known to have low-magnitude inputs, and power gains should be measured using simulation.

-tp_opt_tree auto | true | false

Controls the usage of carry-save adder tree for low power optimization based on transition probabilities (TP). This optimization reduces internal switching activity and therefore the dynamic power in SOPs when the inputs have a non-uniform transition

probability distribution.

When you set the **-tp_opt_tree** option to **auto**, the default, the tool uses the option only if it determines a QoR benefit and optimized dynamic power. When you set the **-tp_opt_tree** option to **true**, the tool always uses the option if dynamic power is optimized. When you set the **-tp_opt_tree** option to **false**, the tool never uses the option.

-tp_oper_sel auto | true | false

Controls the selection or ordering of multiplier input operands for low power based on transition probability (TP). This optimization can reduce internal switching activity and therefore dynamic power in booth multipliers when the inputs have significantly different transition probabilities.

When you set the **-tp_oper_sel** option to **auto**, the default, the tool uses the option only if it determines a QoR benefit and optimized dynamic power. When you set the **-tp_oper_sel** option to **true**, the tool always uses the option if dynamic power is optimized. When you set the **-tp_oper_sel** option to **false**, the tool never uses the option. The option only affects multipliers with booth encoding.

-inv_out_adder_cell auto | true | false

Controls the usage of full-adder cells with inverted sum-and-carry outputs in the target library for compilation.

When you set the **-inv_out_adder_cell** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-inv_out_adder_cell** option to **true**, the tool always uses the option. When you set the **-inv_out_adder_cell** option to **false**, the tool never uses the option.

This option may not work if the special cells are not availabe in the library.

-4to2_compressor_cell auto | true | false

Controls the usage of 4-2 compressor cells in the target library for compilation.

When you set the **-4to2_compressor_cell** option to **auto**, the default, the tool uses the option only if it determines there is a QoR benefit. When you set the **-4to2_compressor_cell** option to **true**, the tool always uses the option. When you set the **-4to2_compressor_cell** option to **false**, the tool never uses the option.

This option may not work if the special cells are not availabe in the library.

-optimize_for auto | area | speed | area,speed

Supplies a specific optimization goal for the specified cells.

When you set the **-optimize_for** option to **auto**, the default, the tool chooses the optimization mode based on QoR benefit. When you set the **-optimize_for** option to **area**, **speed** or **area,speed**, the tool always uses the specified optimization mode during compile.

-mult_arch *architecture*

Specifies the multiplier architecture to be implemented. This option is a concise representation for a subset of six architecture options. Valid values are **auto**, **and**, **nand**, **and_radix4**, **nand_radix4**, **benc_radix4**, **benc_radix8**, **benc_radix4_mux**, and **benc_radix8_mux**.

The different **-mult_arch** options translate to the following combinations of generation settings:

and: AND-based non-booth architecture -booth_encoding false -mult_radix4 false -mult_nand_based false

nand: NAND-based non-booth architecture -booth_encoding false -mult_radix4 false -mult_nand_based true

and_radix4: AND-based radix-4 non-booth architecture -booth_encoding false -mult_radix4 true -mult_nand_based false

nand_radix4: NAND-based radix-4 non-booth architecture -booth_encoding false -mult_radix4 true -mult_nand_based true

benc_radix4: radix-4 booth architecture -booth_encoding true -booth_radix8 false -booth_mux_based false

benc_radix8: radix-8 booth architecture -booth_encoding true -booth_radix8 true -booth_mux_based false

benc_radix4_mux: MUX-based radix-4 booth architecture -booth_encoding true -booth_radix8 true -booth_mux_based true

benc_radix8_mux: MUX-based radix-8 booth architecture -booth_encoding true -booth_radix8 true -booth_mux_based true

module_or_cell_list

An optional list of cells or modules. When present, the switch settings in this command affect only the specified objects. Otherwise the settings are applied on the whole design.

If you set datapath architecture options on a module and an instance that instantiates this module, the setting on the instance is used by the generator.

The option settings are always hierarchical. All datapath generation will use the same options under the hierarchy of specified objects. If the options are set on different objects in a hierarchy, the tool will use the settings of the closest object to the datapath cell. It will check the options of the cell itself, then the reference module of this cell, then the parent cell, and the reference module of the parent cell, and so on, till the top module. The global options on the whole design is the last one to check.

DESCRIPTION

The **set_datapath_architecture_options** command controls datapath generation strategies used in datapath synthesis.

The default value for all datapath architecture strategies is **auto**. By default, the best strategies are automatically chosen based on the given design constraints. Setting the options to **auto** is equivalent to not setting anything at all, and can be used to clean the existing settings.

When the datapath architecture options are specified without an object list, the specified options are automatically applied to the entire design. When the datapath architecture options are specified with an object list, the specified options are applied only on the specified objects. The object can be a module, a hierarchical cell, or a synthetic operator. The options do not take effect when they are applied on the generic GTECH cell or the cell that is already mapped to the technology library.

When the options are applied on a synthetic operator, and if the operator is extracted into DP_OP cell, the options will be merged with other operator's options and inherited to the DP_OP cell. All operators within this DP_OP cell will use the same options on this DP_OP cell.

When synthetic operators with different option values are extracted into a DP_OP cell, the DP_OP cell merges the option value based on the following priority: For the -optimize_for option, the option value **speed** has the highest priority, **area,speed** is the next, and **area** has the lowest priority. For the options take **true** and **false**, the option value **true** has the highest priority. **false** is the next, and **auto** has the lowest priority.

The synthetic operators could go through multiple complex transformations during compile_ultra. For example, two operators may be shared at the beginning and unshared later depending on the design constrains. When the synthetic operator with the datapath architecture options go through such transformations, it is possible that the initial option setting is not preserved.

When you specify the architecture options on the same object the second time, the new option values will be appended to existing options if no conflict, or overwrite the old values if conflict happens.

The option values set by set_datapath_architecture_options command can be checked by **get_attribute** command with attribute name "dpgen_options". The options can be cleaned by **remove_attribute** command, or by specifying **-all_options auto** with **set_datapath_architecture_options** command.

The **report_dpgen_options** command can report all architecture options user has ever set, including the ones set on module, cell, and the whole design, before compile. After compile, if the objects having architecture options are not ungrouped, their architecture options can also be reported by this command.

The datapath architecture options are also listed in the datapath optimization report by report_resources command. This is the options the tool used during compile, and may be different from what was initially set, due to complex datapath transformations.

EXAMPLES

The following example uses **set_datapath_architecture_options** to control the usage of booth encoded architectures when generating multipliers in current design.

```
prompt> set_datapath_architecture_options -booth_encoding true
```

The following example sets all the options to **auto**, except for **-booth_encoding**, which is set to **true** and overrides the value of **-all_options**:

```
prompt> set_datapath_architecture_options -all_options auto -booth_encoding true
```

The following example append the **-4to2_compressor** option to previous options. The final option value on U1 is **-booth_encoding true -booth_radix8 true -booth_mux_based false -4to2_compressor_cell true**.

```
prompt> set_datapath_architecture_options -mult_arch benc_radix8 [get_cells U1]
prompt> set_datapath_architecture_options -4to2_compressor_cell true [get_cells U1]
```

In following example, when combined with other datapath architecture options related to multiplier architecture, the **-mult_arch** option takes precedence. The final options is **-booth_encoding true -booth_radix8 true -booth_mux_based false.**

```
prompt> set_datapath_architecture_options -mult_arch benc_radix8 -booth_encoding false
```

SEE ALSO

get_attribute(2)
remove_attribute(2)
report_dpgen_options(2)
report_resources(2)

set_datapath_gating_options

Controls the gating behavior when generating the datapath cell.

SYNTAX

status **set_datapath_gating_options**
[-enable true | false]
[-sequential true | false]
[-ungroup true | false]
[-reset]
[module_or_cell]

Data Types

module_or_cell list

ARGUMENTS

-enable true | false

Enable or disable datapath gating. True by default.

When you set the **-enable** option to **true**, the tool will do datapath gating if applicable, which is the default behavior. When you set the **-enable** option to **false**, the tool won't do datapath gating.

-sequential true | false

Enable or disable sequential datapath gating. False by default.

When you set the **-sequential** option to **true**, the tool will do the sequential datapath gating if applicable. When you set the **-sequential** option to **false**, the tool won't do sequential datapath gating, which is the default.

This option can only be set in global level. It can't be followed by modules or cells. This option is ignored if **-enable** is **false**.

-ungroup true | false

Specifies if the gated datapath cell should be ungrouped or not. True by default.

When you set the **-ungroup** option to **true**, the tool will always ungroup the gated datapath cell, which is the default behavior. When you set the **-ungroup** option to **false**, the tool won't ungroup the gated datapath cell.

This option takes effect only when the datapath cell is gated.

-reset

Reset all settings ever made on current design, including global, module, and instance level settings.

DESCRIPTION

The **set_datapath_gating_options** command sets an attribute on the list of specified modules or instances, or current design if no targets are specified. The attribute controls whether or not the datapath cells in a design is optimized by the datapath gating algorithm during the compile, and it can also control the gating options such as ungroup.

By default, the tool will gate or not gate a datapath, based on the power and timing trade off. Datapath internal gating will be attempted if applicable. Sequential datapath gating will be attempted only if it's explicitly turned on.

When the datapath gating options are specified without an object list, the specified options are automatically applied to the entire design. When the datapath gating options are specified with an object list, the specified options are applied only on the specified objects. The object can be a module, a hierarchical cell, or a synthetic operator. The options do not take effect when they are applied on the generic GTECH cell or the cell that is already mapped to the technology library.

Datapath gating related optimizations such as output selector extraction, sequential gating based port punching etc. can only be enabled or disabled when this command is called in global level. Datapath gating itself can be enabled or disabled in instance level, module level, or global level.

When the options are applied to a synthetic operator, and if the operator is extracted into DP_OP cell, the options will be merged with other operator's options and inherited to the DP_OP cell. All operators within this DP_OP cell will use the same options on this DP_OP cell.

When synthetic operators with different option values are extracted into one DP_OP cell, the DP_OP cell merges the option value based on the following priority: For the options take **true** and **false**, the option value **true** takes the highest priority.

The synthetic operators could go through multiple complex transformations during compile_ultra. For example, two operators may be shared at the beginning but unshared later depending on the design constrains. When the synthetic operator with the datapath gating options go through such transformations, it is possible that the initial option setting is not preserved.

When you specify the gating options on the same object the second time, the new option values will be appended to existing options if no conflict, or overwrite the old values if conflict happens.

When gating option is set on different level of hierarchy, the "closest" option to the synthetic instance will be used during datapath generation. By other words, the generator checks settings in the order of instance, parent instance, parent module, so on and so forth till the top design.

The **report_datapath_gating_options** command can report all gating options user has ever set, including the ones set on module, cell, and the whole design, before compile. After compile, if the objects having gating options are not ungrouped, their gating options can also be reported by this command.

The datapath gating options are also listed in the datapath optimization report by report_resources command. This is the options the tool used during compile, and may be different from what was initially set, due to complex datapath transformations.

EXAMPLES

In the following example, datapath gating is disabled on the datapath cells pertained to U4 and U5.

```
prompt> set_datapath_gating_options -enable false [get_cells {U4 U5}]
```

The following example specifies that the U3/U5 instance should not be ungrouped if it's gated by the tool:

```
prompt> set_datapath_gating_options -ungroup false [get_cells U3/U5]
```

The following example shows the cumulative behavior of the command. The datapath gating is disabled globally, then enabled for hierarchical cell **U1**/**U3**, later additionally disable the ungrouping for **U1**/**U3**/**add_8**. The effect of this is that all datapath blocks under U1/U3 will be eligible for gating, while others in the design won't be gated, and specially the datapath containing **U1**/**U3**/**add_8** won't be ungrouped (or **U1**/**U3**/**add_8** won't be ungrouped if not extracted to a datapath block) if it's actually gated.

```
prompt> set_datapath_gating_options -enable false
1

prompt> set_datapath_gating_options -enable true [get_cells U1/U3]
1

prompt> set_datapath_gating_options -enable true -ungroup false [get_cells U1/U3/add_8]
1
```

Th following example removes all previous datapath gating settings:

```
prompt> set_datapath_gating_options -reset
1
```

SEE ALSO

report_datapath_gating_options(2)

set_db_file_mapping

Creates a mapping between original input .db files(built into the reference ndm), and the new .db files to be used during PrimeTime delay calculation.

SYNTAX

string **set_db_file_mapping**
[-library *library*]
original_db_file_name
current_db_file_name

Data Types

library	collection
original_db_file_name	string
current_db_file_name	string

ARGUMENTS

-library *library*

Specifies the lib-cell library that has been created using the original DB file. If this is not specified, objects will be found in the current design.

original_db_file_name

Specifies the base name or full path name input DB file that has been used to generate the specified library.

current_db_file_name

Specifies the base name or full path name DB file that has been used to perform delay calculation.

DESCRIPTION

Create a mapping between original input .db files (which built into the reference ndm), and the new .db files to be used during PrimeTime delay calculation. When PrimeTime delay calculation enabled, the tool will try to find the same set of .db file names from the search path. If a different set of .db file names should be used now, you can use this command to map the old .db file names to the new ones.

The feature is supported when PrimeTime delay calculator is enabled or the **analyze_timing_correlation** command is used.

When the .db mapping is specified using this command, the mapped .db file name is searched using the following rules:

- If the mapping file uses full path for the new db, then the tool will just use the full path to locate the new .db
- If the mapping file use simple name for the new db, then the tool will search the db from the specified search_path.
- If the mapping file use relative path for the new .db file , then the tool will search the .db from the search_path/"relative path specified in the mapping".

EXAMPLES

The following example creates a mapping between input DB file and the current mapped DB file.

```
prompt> set_db_file_mapping /A/B/C/1.db    /E/F/G/1.db
```

SEE ALSO

set_density_gradient_options

Sets options for **report_placement -hard_macro_density_gradient_violations** command.

SYNTAX

status **set_density_gradient_options**
[-white_space_density {*layer_name_and_density_pairs*}]
[-edge_zone_width {*layer_name_and_inner_width_pairs*}]
[-outer_zone_width {*layer_name_and_outer_width_pairs*}]
[-gradient_tolerance {*layer_name_and_tolerance_value_pairs*}]
[-min_macro_size *min_macro_size*]
[-cluster_threshold *cluster_threshold*]

Data Types

layer_name_and_density_pairs list
layer_name_and_inner_width_pairs list
layer_name_and_outer_width_pairs list
layer_name_and_tolerance_value_pairs list
min_macro_size float
cluster_threshold float

ARGUMENTS

-white_space_density {*layer_name_and_density_pairs*}

Specifies the white space density. The white space density is measured at areas without metal density definition inside macros and areas outside of macros. Specify a list of layer names and white space density pairs in the curly braces. The layer name must be a string, such as OD, M1, M2 and so on. The data type of the white space density is float and the unit is micron. The pair format is:

{layer_name1 density1 layer_name2 density2 layer_name3 density3 ...}

By default, the command uses a density of 0.35 as the white space density for each layer. You can specify white space density from 0 to 1.

-edge_zone_width {*layer_name_and_inner_width_pairs*}

Specifies the width used to form the inner window inside the hard macro. The width is the distance extending inwards from the macro edge, and this distance along with the length of the macro edge forms the inner window. If the inner window extends outside the bbox of the hard macro, the window stops at the bounding box of the hard macro. The **report_placement -hard_macro_density_gradient_violations** command calculates the density of inner window for each layer. You can specify a list of layer name and edge zone width pairs. The layer name must be a string, such as OD, M1, M2 and so on. The data type of the width is float and the unit is micron. The pair format is:

{layer_name1 width1 layer_name2 width2 layer_name3 width3 ...}

By default, the command uses 150 microns as edge zone width for each layer.

-outer_zone_width {*layer_name_and_outer_width_pairs*}

Specifies the width used to form the outer window outside the hard macro. The width is the distance extending outward from the macro edge, and this distance along with the length of the macro edge forms the outer window. If the outer window extends outside the chip boundary, the window ends at the chip boundary. The **report_placement -hard_macro_density_gradient_violations** command calculates the density of outer window for each layer. You can specify a list of layer name and outer zone width pair. The layer name must be a string, such as OD, M1, M2 and so on. The data type of the width is float and the unit is micron. The pair format is:

{layer_name1 width1 layer_name2 width2 layer_name3 width3 ...}

By default, the command uses 150 microns as outer zone width for each layer.

-gradient_tolerance {*layer_name_and_tolerance_value_pairs*}

Specifies the gradient tolerance. If the density difference between inner window and outer window is larger than the gradient tolerance, the **report_placement -hard_macro_density_gradient_violations** command reports a violation. You can specify a list of layer name and gradient tolerance pair. The layer name must be a string, such as OD, M1, M2 and so on. The data type of the gradient tolerance is float and the unit is micron. The pair format is:

{layer_name1 value1 layer_name2 value2 layer_name3 value3 ...}

By default, the command uses 0.3 as gradient tolerance for each layer. You can specify gradient tolerance value from 0 to 1.

-min_macro_size *min_macro_size*

Specifies the minimum macro size for checking. If both the width and height of a hard macro are less than the minimum macro size, the **report_placement -hard_macro_density_gradient_violations** command skips checking this hard macro. Also during hard macro edge checking, if the length of an edge is less than half of minimum macro size, the **report_placement -hard_macro_density_gradient_violations** command skips checking this edge. The data type of minimum macro size is float and the unit is micron. By default, the command uses 500 as minimum macro size.

-cluster_threshold *cluster_threshold*

Specifies the cluster threshold. If the hard macros are placed within the distance cluster_threshold, the **report_placement -hard_macro_density_gradient_violations** command merges these hard macros into a cluster. The cluster is treated as a single hard macro for density gradient violations checking. The data type of cluster threshold is float and the unit is micron. By default, the command uses 5 as cluster threshold.

DESCRIPTION

This command sets checking options for the **report_placement -hard_macro_density_gradient_violations** command. This command is additive and can be run multiple times to set the options incrementally.

EXAMPLES

The following example shows the usage of this command.

```
prompt> set_density_gradient_options -white_space_density {M1 0 M2 0.4} \
```

```
-edge_zone_width  {M1 50 M2 60} \
-outer_zone_width {M1 50 M2 60} \
-gradient_tolerance{M1 0.3 M2 0.5} \
-min_macro_size 500 \
-cluster_threshold 5
```

SEE ALSO

report_density_gradient_options(2)
report_placement(2)

set_design_attributes

Sets the specified attributes and their value on required cells.

SYNTAX

string **set_design_attributes**
[-elements *element_list*]
[-attribute *name_value_pair*]
[-models *model_list*]
[-exclude_elements *exclude_list*]
[-is_hard_macro *true*|*false*]
[-is_soft_macro *true*|*false*]

Data Types

element_list list
name_value_pair string
model_list list
exclude_list list

ARGUMENTS

-elements *element_list*

Specifies the collection of cells on which the attributes must be set.

-attribute *name_value_pair*

Specifies the name and value of the attribute to be set on the cells specified by the **-elements** option.

This option can be repeated multiple times to specify multiple attributes for the same set of cells in a single command.

-models *model_list*

Specifies the collection of models on which the attributes must be set.

-exclude_elements *exclude_list*

Specifies a list of elements to be excluded from the effective element list. This option is read and ignored by implementation tools. This option is only valid for attribute ret_mode, in other case, implementation tools drop this command and do not look at the **-elements**, **-models**, and **-exclude_elements** options, This option can be preserved in UPF' or UPF" files only if it is used with attribute ret_mode.

-is_hard_macro *true*|*false*

Specifies models in model_list to be hard-macro models or not. This option must be specified together with **-models** option.

-is_soft_macro *true*|*false*

Specifies if models in model_list are soft-macro models. This option must be specified together with **-models** option.

DESCRIPTION

This command sets the attributes and their value on a list of cells specified by the **-elements** or **-models** options.

Attributes can be set multiple times on the same cells. The last value prevails.

It is an error if the design attribute is specified with neither the **-elements** nor the **-models** options, except for the following design attributes: correlated_supply_group upf_chip_design

The supported attribute names and their associated values are as follows:

- **correlated_supply_group**
  This attribute can be set at the top scope of the design or on hierarchical instance of the design. The value must be one or more groups of supply net names bracketed by curly braces {}. The value can also be the wildcard character (*) alone, which means all supply nets in the design.

  This attribute specifies supply nets whose port state triplets and power state triplets should be considered correlated voltages. The tool considers only the minimum with minimum voltages, only nominal with nominal voltages, and only maximum with maximum voltages, without mixing between minimum, nominal, and maximum. This attribute only accepts dot (.) or a single hierarchical instance as the value for the **-elements** option.

- **derived_feedthrough_hierarchy**
  This attribute can be set on hierarchical cells of the design. The value must be **true** or **false**. The default of this attribute is **false**.

- **derived_iso_strategy**
  This attribute can be set on cell instances of the design. The value must be the name of an isolation strategy.

  This attribute specifies the isolation strategy name to ensure a unique name for the derived strategies in the power domain. It is used in a hierarchical flow to support location fanout.

- **enable_bias**
  This attribute can be set on the top design and also on any cell in the design. The value must be **true** or **false**. The default of this attribute is **false**.

- **enable_state_propagation_in_add_power_state**
  This attribute can be set at the top scope of the design and also on any hierarchical cell. The value must be **true** or **false**. The default of this attribute is **false**.

  When this attribute is set to **true**, the tool propagates the name of the supply set state (specified in the **add_power_state** command) to the functional net. After the state name is propagated to the functional net, you can use the net and the supply set state in the **create_pst** and **add_pst_state** commands. When the value is **true**, usage of the "&&" expression is not allowed in the **supply_expr** option of the **add_power_state** command.

  When this attribute is set to **false**, the supply set state name is not propagated to the functional nets. The state name specified in the **add_power_state** command is a property of the supply set, so the name is not propagated to the functional nets. To create power state tables, you must create GROUPS using the **create_power_state_group** command and refer to the supply set state names in the GROUP. You can no longer refer to the state names in the **add_pst_state** command. When the value is **false**, usage of the "&&" expression is allowed.

  The allowed attribute configurations are:

1. Attribute is set to **false** for the entire design
2. Attribute is set to **true** for the entire design
3. Attribute is set to **true** for the block, and set to **false** for the top design

The following configuration is NOT allowed: Attribute is set to **false** for the block, and set to **true** for the top design.

- **lower_domain_boundary**
  This attribute can be set at the top scope of the design and also on any hierarchical cell. The value must be **true** or **false**.

  When this attribute is set to **true**, the tool considers the domain boundary between a domain and another domain that is contained in a lower-level block as the scope of the first (higher hierarchical level) domain.

- **merge_domain**
  This attribute can be set on cell instances of the design. The value must be **true** or **false**.

  If set to **true**, this attribute implies that the elements that belong to the same power domain can be merged. The cells specified in the **-elements** option should not be the root cell of a power domain.

- **shared_voltage_area**
  This attribute can be set at the top scope of the design or on hierarchical instance of the design. The value must be one or more groups of power domain names bracketed by curly braces {}. This attribute specifies power domains that share a primary voltage area. This attribute only accepts dot (.) as the value for the **-elements** option.

- **SNPS_default_power_upf2sv_vct** and **SNPS_default_ground_upf2sv_vct**
  These attributes specify the default for the UPF-to-SystemVerilog value conversion table. These attributes can be used with the **-models** option.

- **SNPS_default_power_upf2vh_vct** and **SNPS_default_ground_upf2vh_vct**
  These attributes specify the default for the UPF-to-VHDL value conversion table. These attributes can be used with the **-models** option.

- **SNPS_default_power_sv2upf_vct** and **SNPS_default_ground_sv2upf_vct**
  These attributes specify the default for the SystemVerilog-to-UPF value conversion table. These attributes can be used with the **-models** option.

- **terminal_boundary**
  This attribute can be set at the top scope of the design and also on any cell in the list of **-elements** of the command **create_power_domain**. Typically, it is set on a block design, black box, or ETM cell. The value must be **true** or **false**.

  When this attribute is set to **true**, the tool considers the boundary of the cell as a terminal and checks, in the **check_mv_design** command, that the driver/load pin's related supply is consistent with the driver/receiver supply defined on the boundary pin (either by **set_port_attributes -driver_supply -receiver_supply** or by **set_related_supply_net**).

- **upf_chip_design** (used on the top-level design)
  This attribute can be set on the top-level design. The value must be **true** or **false**. It is used in the isolation strategy definition to support location fanout.

- **legacy_block** (used on the top-level design and on the hierarchical instances)
  This attribute can be set on the top-level design and on hierarchical instances. The value must be **true** or **false**. It is used defined if the design or an instance belongs to a legacy block.

- **upf_reconcile_boundary** (used on the top-level design)
  This attribute can be set on the top-level design and block level design. The value must be **skip** or **reconcile_voltages**. It is used in the PSTs merging. The tool will recognize the block UPF PST to be skipped entirely or supply voltages can be reconciled within threshold defined by set_variation command for the supply.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **correlated_supply_group** attribute to specify correlated voltage range comparison for port state triplets.

If a driver is powered by VDD1 and its load is powered by VDD2, a level shifter is not required because VDD1 and VDD2 are in the same correlated supply group and therefore treated as correlated voltage range during voltage comparison.

However, if the load is powered by VDD3, a level shifter is required because VDD1 and VDD3 are not in the same correlated supply group and therefore treated as independent voltage range during voltage comparison.

```
prompt> add_port_state VDD1 -state {HV 0.8 1.0 1.2}
1
prompt> add_port_state VDD2 -state {HV 0.8 1.0 1.2}
1
prompt> add_port_state VDD3 -state {HV 0.8 1.0 1.2}
1
prompt> add_port_state VDD4 -state {HV 0.8 1.0 1.2}
1
prompt> set_design_attributes \
  -elements {.} \
  -attribute correlated_supply_group "{VDD1 VDD2} {VDD3 VDD4}"
1
```

The following example sets the **SNPS_default_power_sv2upf_vct** attribute SV_LOGIC2UPF on model bot and libcell GTECH_NOT.

```
prompt> set_design_attributes \
  -models {bot GTECH_NOT} \
  -attribute SNPS_default_power_sv2upf_vct SV_LOGIC2UPF
1
```

The following example sets the **upf_chip_design** attribute on a top-level design.

```
prompt> set_design_attributes -elements {.} \
  -attribute upf_chip_design true
1
```

The following example sets the **merge_domain** attribute to **true** on cell mid1.

```
prompt> set_design_attributes \
  -elements {mid1} \
  -attribute merge_domain true
1
```

The following example sets the **derived_iso_strategy** attribute on mid11 with power domain MID11.

```
prompt> set_design_attributes \
  -elements {mid11} \
  -attribute derived_iso_strategy iso1
1
prompt> set_isolation iso1 \
  -domain MID11 -source sset1 \
```

```
  -elements {mid11/*} \
  -isolation_supply_set sset3
1
prompt> set_isolation_control iso1 \
  -domain MID11 \
  -isolation_signal in1 \
  -location fanout
1
```

The following example sets the **lower_domain_boundary** attribute to TRUE on the top scope and cell mid1.

```
prompt> set_scope
1
prompt> set_design_attributes /
  -elements {.} /
  -attribute lower_domain_boundary TRUE
Information: Resolving '.' to the current scope 'top'.
1
prompt> set_design_attributes /
  -elements {mid1} /
  -attribute lower_domain_boundary TRUE
1
```

The following example sets the **enable_state_propagation_in_add_power_state** attribute to **false** on the top scope and to **true** on cell mid1.

```
prompt> set_scope
1
prompt> set_design_attributes \
  -elements {.} \
  -attribute enable_state_propagation_in_add_power_state false
1
prompt> set_design_attributes \
  -elements {mid1} \
  -attribute enable_state_propagation_in_add_power_state true
1
```

The following example sets the **terminal_boundary** attribute to **true** on the top scope and cell mid1.

```
prompt> set_scope
1
prompt> set_design_attributes \
  -elements {.} \
  -attribute terminal_boundary true
1
prompt> set_design_attributes \
  -elements {mid1} \
  -attribute terminal_boundary true
1
```

The following example sets the **upf_reconcile_boundary** attribute to on the IP block.

```
prompt> set_design_attributes \
  -models {IP_A} \
  -attribute upf_reconcile_boundary skip
1
prompt> set_design_attributes \
  -models {IP_B} \
```

-attribute upf_reconcile_boundary reconcile_voltages
1

SEE ALSO

set_port_attributes(2)
set_related_supply_net(2)
set_variation(2)

set_design_rule_attribute

Set multiple attributes together for design rule object.

SYNTAX

collection **set_design_rule_attribute**
-list *name_value_list*
[-objects *object_list*]
[patterns]

Data Types

name_value_list	list
object_list	collection
patterns	collection

ARGUMENTS

-list *name_value_list*

Specifies a list of attribute name and value pairs which to set. The list must have an even number of tokens to be valid. Note that if any of the key invalid, that key will be ignored. if any of the value supplied is invalid, this command fails with no effect.

-objects *object_list*

Specifies a collection of design rule objects for which attribute needs to be set. Either one (and only one) of '-objects' or 'patterns' must be specified.

patterns

Specifies a collection of design rule objects for which attribute needs to be set. Either one (and only one) of 'patterns' or '-objects' must be specified.

DESCRIPTION

This command will allow user to set multiple attribute of design rule together. Currently, set_attribute command doesn't have the option to set multiple attributes together. Setting design rule attributes require sanity check on the updated design rule object. So to verify the sanity of updated design rule object, it might possible that multiple attributes update required. Hence this command will be useful in those scenarions.

EXAMPLES

The following example sets multiple attributes of a design rule in the technology object of the current library.

```
prompt> set_design_rule_attribute -list \
   {end_of_line_enc_5_neighbor_width_threshold 2.1 \
   end_of_line_enc_5_neighbor_cut_tbl_size 1 \
   end_of_line_enc_5_neighbor_cut_name_tbl Vs \
   end_of_line_enc_5_neighbor_max_spacing 1.2 \
   end_of_line_enc_5_neighbor_parallel_length 0.1 \
   end_of_line_enc_5_neighbor_extension_range 0.01 \
   end_of_line_enc_5_neighbor_min_enclosure 0.2 \
   end_of_line_enc_5_neighbor_check_length 1.1 \
   end_of_line_enc_5_neighbor_check_width 0.03 \
   end_of_line_enc_5_neighbor_corner_keepout_width 0.036 \
   end_of_line_enc_5_neighbor_min_spacing 0.063 \
   } DESIGN_RULE_0
```

The following example sets single attribute on multiple design rule objects.

```
prompt> set_design_rule_attribute -list {min_spacing 1.1} \
  -objects [get_design_rules DESIGN_RULE*]
```

SEE ALSO

create_design_rule(2)
get_design_rules(2)
remove_design_rules(2)
report_design_rules(2)

set_design_top

Specifies the top-level design instance for simulation and verification tools.

SYNTAX

status **set_design_top**
instance_name

Data Types

instance_name	string

ARGUMENTS

instance_name

Specifies the top-level instance in the design.

DESCRIPTION

The **set_design_top** command specifies the top-level design instance. This information is used only by simulation and verification tools.

EXAMPLE

The following example shows how to use the **set_design_top** command:

```
prompt> set_design_top ALU07
```

set_device_group_type

This command specifies which device group names for library cells are wide, normal, or narrow type.

SYNTAX

string **set_device_group_type**
  -type *device_type*
  group_names

string *device_type*
list *group_names*

ARGUMENTS

-type *device_type*

Specifies the device type for the specified groups. Must be one of **wide**, **normal**, or **narrow**.

group_names

A list of device group names, which are stored as **device_group** attributes on lib_cell objects.

DESCRIPTION

This command specifies which device group names for library cells are wide, normal, or narrow device type.

It is to be noted that device group type information is non persistent

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example specifies that lib_cells with "device_group" attribute equal to "WIDE" or "wide" are considered as "wide" type for optimization.

prompt> **set_device_group_type -type wide {WIDE wide}**

SEE ALSO

set_attribute(2)

set_dft_clock_controller

Specifies the parameters for DFT-inserted clock controllers.

SYNTAX

status **set_dft_clock_controller**
-cell_name the_controller_cell_name
-ateclock the_slow_clock
-test_mode_port the_test_mode
-pllclocks pin_list
[-chain_count num_chain_count]
[-cycles_per_clock num_cycles]

Data Types

the_controller_cell_name string
the_slow_clock string
the_test_mode string
pin_list list
num_chain_count integer
num_cycles integer

ARGUMENTS

-cell_name the_controller_cell_name

Specifies the instance name of the clock controller that will be inserted by the **insert_dft** command.

-ateclock the_slow_clock

Specifies the slow clock used for scan shifting that you want to connect to the DFT clock controller.
The specified port must be defined as a RefClock signal using the set_dft_signal command.

-test_mode_port the_test_mode

Specifies the test mode port used to enable the clock controller.
The specified port must be defined as a TestMode signal with -usage occ using the set_dft_signal command.

-pllclocks pin_list

Specifies the list of PLL output clock pins to control.
The pins must be previously defined as [-type MasterClock -usage occ] signals using set_dft_signal command.

-chain_count num_chain_count

Specifies the number of clock chains to insert per clock controller. If not specified, the default is 1.

-cycles_per_clock num_cycles

Specifies the maximum number of capture cycles per clock.
If not specified, the default is 2.

DESCRIPTION

The **set_dft_clock_controller** command specifies the clock controller characteristics for the **insert_dft** command to insert and connect clock controllers and clock chains into the design.

A clock controller is a design that controls which clock signals will propagate into scan chains during scan shifting and capture. The clock controller selects between fast clocks coming from an on-chip clock generator (usually a phase-locked loop) and slow clocks coming from external ports.

A clock chain is a scan chain segment of one or more scannable control
registers. This chain allows for a per-pattern clock selection mechanism by ATPG. Clock selection values are loaded into the clock chain as part of the regular scan load process.

The clock controller cell must be included in a clock chain using the command:

set_scan_path the_occ_chain_name -include [list the_controller_names] -class occ

The number of the [set_scan_path -class occ] commands will determine the number of clock chains of the design.

EXAMPLES

```
prompt>set_dft_signal -type ScanDataIn -port OCC_SI
prompt>set_dft_signal -type ScanDataOut -port OCC_SO
prompt>set_dft_signal -type RefClock -port ate_clk -timing [list 45 55]
prompt>set_dft_signal -type MasterClock -hookup_pin pll_1/pll_clk -usage occ
prompt>set_dft_signal -type MasterClock -hookup_pin pll_2/pll_clk -usage occ
prompt>set_dft_signal -type testmode -port occ_test_mode -usage occ
prompt>set_dft_clock_controller -cell_name occ_ctrl -ateclock ate_clk -pllclocks {pll_1/pll_clk pll_2/pll_clk } -test_mode
occ_test_mode
prompt>set_scan_path my_occ_chain -scan_data_in OCC_SI -scan_data_out OCC_SO -include {occ_ctrl} -class occ
```

SEE ALSO

set_scan_path(2)
preview_dft(2)
insert_dft(2)

set_dft_clock_gating_configuration

Specifies the clock-gating configuration for a design.

SYNTAX

status **set_dft_clock_gating_configuration**
[-exclude_elements *object_list*]

Data Types

object_list cells, lib_cells, or clocks

ARGUMENTS

-exclude_elements *object_list*

Specifies a list of clock-gating cells, clock-gating observation cells, hierarchical cell instances containing these cells, lib_cells of these cells, or clocks of these cells that are to be excluded when the test control pins of clock-gating cells are connected during DFT insertion. These excluded clock-gating cells are not checked for unconnected test pins during **dft_drc**, and therefore they do not cause TEST-130 violations. Downstream registers driven by these excluded clock-gating cells will have DRC violations if their clocks are uncontrollable.

If a hierarchical cell instance is specified, all clock-gating cells and clock-gating observation cells in that instance are included in the specification.

If a clock-gating library cell is specified, all instances of that cell are included in the specification.

If a clock is specified, clock-gating cells in the respective clock domain are included in the specification. The clock must be defined with the **create_clock** command.

Cells with CTL models or inside CTL models are ignored and excluded from the specification.

This option accepts wild cards and collection inputs. This option is cumulative across multiple commands.

DESCRIPTION

This command specifies the clock-gating configuration for DFT flows. It affects how clock-gating logic is handled by the **dft_drc**, **insert_dft**, and **compile** commands.

Exclusion works by leaving the default deasserted constant value connected to the test pins of excluded clock-gating cells. If an exclusion applies to a clock-gating cell with an existing test pin connection other than the default deasserted constant value, that existing test pin connection remains in place.

The precedence of clock-gating cell test pin connection control methods is as follows, in order of highest to lowest priority:

- **set_dft_signal -connect_to** (highest priority)
- **set_dft_clock_gating_configuration -exclude_elements**

The **-exclude_elements** exclusion option will function with, but is not intended for, registers driven by user-defined clock-gating cells defined via **set_dft_clock_gating_pin** command. If you do not want to connect the test pins of these clock-gating cells, then they should not be defined. Registers driven by user-defined clock gating cells are traced through simple buffer and inverter logic only.

EXAMPLES

The following example excludes a particular clock-gating cell from having its test pin connected:

```
prompt> set_dft_clock_gating_configuration \
    -exclude_elements I_ADD/clk_gate_add_out_reg
```

The following example excludes clock-gating cells and clock-gating observation logic inside a hierarchical cell instance:

```
prompt> set_dft_clock_gating_configuration -exclude_elements U_block
```

The following example excludes all instances of a particular library cell of the 'slow' library:

```
prompt> set_dft_clock_gating_configuration \
    -exclude_elements [get_lib_cell slow/ABCD123XYZ]
```

The following example excludes a particular clock-gating observation leaf cell:

```
prompt> set_dft_clock_gating_configuration -exclude_elements Uclk_gate_obs
```

The following example excludes clock-gating cells in a particular clock domain:

```
prompt> create_clock -period 5 CLK

prompt> set_dft_clock_gating_configuration -exclude_elements CLK
```

set_dft_clock_gating_pin

Specifies the test pin of a clock-gating cell in a design. The main purpose of this command is to identify the unconnected test pins of the clock-gating cells that were not inserted by Power Compiler. These pins are connected to test ports when you run DFT insertion.

SYNTAX

status **set_dft_clock_gating_pin**
object_list
-pin_name *instance_pin_name*
[-control_signal ScanEnable | TestMode]
[-active_state 1 | 0]

Data Types

object_list list

ARGUMENTS

object_list

Specifies the clock-gating cell instances for which test pins are specified. This is a required argument.

Wildcards and collections are supported.

-pin_name *instance_pin name*

Specifies the name of the test pin on the specified instances. This pin name is common to all specified instances. This is a required argument.

-control_signal ScanEnable | TestMode

Specifies the type of control signal required by the test pin. DFT insertion connects the pin to a test port of the specified type. This argument is optional. By default, the control signal type is ScanEnable.

-active_state 1 | 0

Specifies the active state of the test pin. This argument is optional. By default, the active state is high, that is, 1.

DESCRIPTION

Specifies the test pins of clock-gating cells not inserted by Power Compiler for DFT flows. The command affects how clock-gating logic is handled by the **dft_drc**, **insert_dft**, and **compile** commands.

Identified cells can be connected to user-specified test ports by using the **set_dft_signal -connect_to** command. However, they are not connected when clock-domain-based connection is done, by using the **set_dft_signal -connect_to** command. For clock-domain-based connection, only clock-gating cells recognized and inserted by Power Compiler are supported.

You must be careful when specifying the instance pin name because the test pin name is common to all specified instances.

There is no checking for the specified cells. It is your responsibility to make sure that the specified cell and pin are actually a clock-gating cell and test pin that was not identified by Power Compiler. Specifying a cell that is not a clock gating cell can cause undesired results when you run DFT insertion.

Existing connections are maintained. If the specified test pin is already connected to a test port or logic, the tool command does not disconnect and reconnect the test pin. Also, the **dft_drc** command will not issue a TEST-130 message for the pin. Only test pins driven by a logic constant and not connected to a net or to a net without a driver are handled by the **dft_drc** command and subsequently connected by the tool.

EXAMPLES

The following examples identify a particular clock-gating cell.

```
# Identification using the default setting:

prompt> set_dft_clock_gating_pin {sub1/clk_gate_out1_reg} \
        -pin_name TE

# Identification using explicit specifications:

prompt> set_dft_clock_gating_pin {sub1/clk_gate_out1_reg} \
        -pin_name TE -control_signal ScanEnable -active_state 1

prompt> set_dft_clock_gating_pin {sub2/clk_gate_out1_reg} \
        -pin_name TE -control_signal ScanEnable -active_state 0

# Identification using explicit specifications with the test pin controllable by a test mode:

prompt> set_dft_clock_gating_pin {sub3/clk_gate_out1_reg} \
        -pin_name TE -control_signal TestMode

prompt> set_dft_clock_gating_pin {sub4/clk_gate_out1_reg} \
        -pin_name TE -control_signal TestMode -active_state 0

# Identify test pin SE on all instances of library cell MyCGCell with default settings:

prompt> set_dft_clock_gating_pin [get_cells * -hierarchical \
        -filter "ref_name == MyCGCell"] -pin_name SE
```

SEE ALSO

dft_drc(2)
insert_dft(2)
report_dft(2)

set_dft_configuration

Sets the DFT configuration for the current design.

SYNTAX

status **set_dft_configuration**
[-scan enable | disable]
[-bsd enable | disable]
[-ait enable | disable]
[-hsio enable | disable]
[-wrapper enable | disable]
[-clock_controller enable | disable]
[-scan_compression enable | disable]
[-pipeline_scan_data enable | disable]
[-testability enable | disable]
[-logicbist enable | disable]
[-ieee_1500 enable | disable]
[-mode_decoding_style binary | one_hot]

ARGUMENTS

-scan enable | **disable**

Enables or disables the SCAN utility.

The default is **enable**.

-bsd enable | **disable**

Enables or disables the insertion of IEEE Std 1149.1/1149.6 compliant boundary-scan circuitry.

-ait enable | **disable**

Enables or disables the insertion of AIT Controller.

-hsio enable | **disable**

Enables or disables the insertion of HSIO Controller.

-wrapper enable | **disable**

Enables or disables the Core wrapper utilities. The specifications for the utilities are set with the **set_wrapper_configuration** command.

-clock_controller enable | **disable**

Enables or disables the clock controller insertion for on chip clocking.

-scan_compression enable | disable

Enables or disables the insertion of scan compression structures.

-pipeline_scan_data enable | disable

Enables or disables the insertion of pipeline scan data registers to scan chains. By default, the pipeline scan data utility is disabled.

-testability enable | disable

Enables or disables the testability test-point client (automatic test-point insertion and user-defined test points).

-logicbist enable | disable

Enables or disables the insertion of LogicBIST compression, which provides built-in self-test capability for scannable logic.

-ieee_1500 enable | disable

Enables or disables the IEEE 1500 test-mode control functionality.

Use this option to enable both the DFT-inserted IEEE 1500 controller flow and the existing IEEE 1500 controller flow.

-mode_decoding_style binary | one_hot

Specifies the type of port decoding to be used for the test modes controller. Allowed values are binary and one_hot. The default value is binary.

DESCRIPTION

The **set_dft_configuration** command specifies the DFT configuration for a design. The DFT configuration is specific to the current design. If you change the DFT configuration and then change the current design, your changes are no longer visible.

Use the **report_dft** command to display the current DFT configuration of the design.

EXAMPLES

The following command disables the stitching of scan chain during compile

prompt> **set_dft_configuration -scan disable**

SEE ALSO

insert_dft(2)
report_dft(2)
set_dft_signal(2)

set_dft_drc_configuration

Sets the DFT DRC configuration for the current design.

SYNTAX

status **set_dft_drc_configuration**
[-internal_pins enable | disable]
[-static_x_analysis enable | disable]
[-occ_bypass enable | disable]
[-pre_dft_drc_verbose_report_in_compile enable | disable]
[-pre_dft_drc_report_file_in_compile *filename*]
[-use_test_model true | false]
[-allow *drc_list*]
[-ignore *drc_list*]
[-allow_se_set_reset_fix true | false]
[-clock_gating_init_cycles *integer*]

Data Types

filename string
drc_list string

ARGUMENTS

-internal_pins

Enables the "internal pins" flow, in which internal pins can be specified as clock, reset, set, constant, scan-in, scan-out, scan-enable, and test_mode signals.

When **-internal_pins** is set to **enable**, you can specify internal pins to be as clock, reset, set, constant, scan-in, scan-out, scan-enable and test_mode signals. You can specify these internal pins using the **set_dft_signal** and **set_scan_path** commands. These signals are then used by the **dft_drc**, **preview_dft**, **insert_dft**, and **compile** commands. However, the protocol generated at the end of post-DFT DRC is not complete and cannot be used by the user.

-static_x_analysis enable | **disable**

Enables pre-DFT DRC to report static-X scan cell violations. The default is **disable**.

-occ_bypass enable | **disable**

Enables the on-chip clocking (OCC) bypass mode during post-DFT DRC.

-pre_dft_drc_verbose_report_in_compile disable | **enable**

Controls whether the pre-DFT DRC report generated by the **compile_fusion** command is reported in summary or verbose form. The default is the summary form.

-pre_dft_drc_report_file_in_compile *filename*

Controls whether the pre-DFT DRC report generated by the **compile_fusion** command is written to a file or included in the compile comand output. The default is to include it in the compile command output.

-use_test_model true | false

Replaces DFT-inserted blocks with their test model representations during DFT DRC and scan architecting. The default is **true**.

When this option is set to **false**, **dft_drc** uses the gates instead of the test model attached to a block. Use this only for blocks containing scan chains and complex test control structures, where the test model does not accurately describe the test control logic.

-allow *drc_list*

Specifies that for the specified DRC violations, DFT should allow violating cells to be included in scan chains. The violations are still reported.

-ignore *drc_list*

Specifies that for the specified DRC violations, DFT should completely ignore the violations. The violating cells are included in scan chains and the violations are not reported.

-allow_se_set_reset_fix true | false

Controls whether pre-DFT and post-DFT DRC allow internally generated reset signals that are gated by the scan-enable signal. When set to the default of **false**, DFT DRC does not allow scan cells to be driven by any internally generated reset; they are marked as violations and excluded from scan chains. When set to **true**, DFT DRC allows scan cells whose internally generated reset signals are gated by the scan-enable signal during scan shift.

Note that scan-enable signals driving existing testability logic in your design must be defined with the **-view existing_dft** option of the **set_dft_signal** command for pre-DFT DRC to consider them, even if they are already also defined with the **-view spec** option.

This option is equivalent to the **set_drc -allow_unstable_set_reset** option in TetraMAX ATPG.

-clock_gating_init_cycles *integer*

Adds the specified number of additional clock pulses to the test_setup section of the test protocol. This is useful for initialization of multiple cascaded sequential clock-gating cells. The default is zero.

DESCRIPTION

This command specifies the configuration for DFT DRC, which is performed explicitly by the **dft_drc** command and can be performed implicitly by the **preview_dft**, **insert_dft**, and **compile** commands if not previously run.

Allowing or Ignoring Certain DRC Violations

The **-allow** and **-ignore** options can be used to change how the following DRC rule violations are treated by DFT insertion:

- TEST-504 (warning) Cell %s has constant 0 value.
- TEST-505 (warning) Cell %s has constant 1 value.
- D17 (warning) D17 Clock input I of <type> S couldn't capture data.

By default, these DRC violations cause DFT to omit the violating cells from scan chains. You can use this command to include them in scan chains, with or without the violations being reported.

These options are similar to the **set_rules** command in TetraMAX in that both commands control how the tool treats DRC rule violations. However, they are different in that these options control how the scan chain hardware is built.

Note that the following commands take precedence over the **-allow** and **-ignore** options:

- If a flip-flop also has a **set_scan_element false** attribute applied, it takes precedence and this specification has no effect.
- If a flip-flop holds a constant value due to the **set_test_assume** command, this assumed constant value takes precedence and this specification has no effect.

EXAMPLES

prompt> **set_dft_drc_configuration -internal_pins enable**

SEE ALSO

dft_drc(2)
set_dft_signal(2)

set_dft_drc_rules

Alters how certain DRC rule violations affect DFT insertion for the specified cells.

SYNTAX

int **set_dft_drc_rules**
[-allow *drc_list*]
[-ignore *drc_list*]
[-cell *cell_list*]

Data Types

drc_list string
cell_list string

ARGUMENTS

-allow *drc_list*

Specifies that for the specified DRC violations, DFT should allow violating cells to be included in scan chains. The violations are still reported.

-ignore *drc_list*

Specifies that for the specified DRC violations, DFT should completely ignore the violations. The violating cells are included in scan chains and the violations are not reported.

-cell *cell_list*

Limits the DFT DRC specification to certain cells. If no cells are specified, the specification applies globally to all cells.

DESCRIPTION

The **set_dft_drc_rules** command can be used to change how certain DRC rule violations affect DFT insertion. The list of DRC rule violation IDs supported by this command is:

- TEST-504 (warning) Cell %s has constant 0 value.
- TEST-505 (warning) Cell %s has constant 1 value.
- D17 (warning) D17 Clock input I of <type> S couldn't capture data.

By default, these DRC violations cause DFT to omit the violating cells from scan chains. You can use this command to include them in scan chains, with or without the violations being reported.

The **set_dft_drc_rules** command in DFT Compiler is similar to the **set_rules** command in TetraMAX in that both commands control how the tool treats DRC rule violations. However, they are different in that the **set_dft_drc_rules** command controls how the DRC violations affect DFT insertion, while the **set_rules** command controls the severity of DRC violations for a static design.

Note that the following commands take precedence over a **set_dft_drc_rules** specification:

- If a flip-flop also has a **set_scan_element false** attribute applied, it takes precedence, and the **set_dft_drc_rules** specification has no effect.
- If a flip-flop holds a constant value due to the **set_test_assume** command, this assumed constant value takes precedence, and the **set_dft_drc_rules** specification has no effect.

EXAMPLES

The following example allows all cells that violate TEST-504 and TEST-505 to be included on scan chains, with the violations reported:

```
prompt> set_dft_drc_rules -allow {TEST-504 TEST-505}
```

The following example allows all cells that violate TEST-504 and TEST-505 to be included on scan chains in the hierarchical instance USPARE_GATES, with the violations not reported:

```
prompt> set_dft_drc_rules -ignore {TEST-504 TEST-505} -cell USPAREGATES
```

SEE ALSO

dft_drc(2)
insert_dft(2)
preview_dft(2)

set_dft_equivalent_signals

Sets a given list of DFT signals as equivalents.

SYNTAX

integer **set_dft_equivalent_signals**
signal_list

Data Types

signal_list list

ARGUMENTS

signal_list

Lists the signals to check against the primary signal for equivalency requirements. The first signal in the list is the primary signal. The primary signal is used to check for valid equivalences with the remaining signals in the list for type and other (if any) equivalency requirements. All the signals must have the same signal type.

DESCRIPTION

This command sets a specified list of scan signals as equivalent for scan architect. This is supported only for clock signal types (ScanClock, MasterClock, and ScanMasterClock), as specified by the **set_dft_signal** command or inferred by the **create_test_protocol** command.

Use this command to specify a set of clocks as equivalent for scan architect. This enables the architect to consider all flops driven by these clocks as if driven by the same clock, and the architect will not add any synchronization elements when they are mixed in a scan chain.

EXAMPLES

The following example illustrates setting two clocks as equivalent to clock clk1:

prompt> **set_dft_equivalent_signals [list clk1 clk2 clk3]**

set_dft_insertion_configuration

Sets the DFT insertion configuration for the current design

SYNTAX

status **set_dft_insertion_configuration**
[-unscan true | false]

ARGUMENTS

-unscan true | **false**

Specifies whether to unscan invalid scan-replaced cells during scan synthesis. A scan-replaced cell is invalid if it has a DRC violation. By default this option is set to false.

DESCRIPTION

The set_dft_insertion_configuration command specifies the configuration to use for the current design during the DFT insertion process. The settings apply only to the current design. If you change the DFT insertion configuration and then change the current design, your changes are no longer visible.

EXAMPLES

```
prompt>set_dft_insertion_configuration -unscan true
```

SEE ALSO

preview_dft(2)
compile(2)
set_scan_configuration(2)

set_dft_isolation

Specify the UPF isolation strategy to be used by dft connections.

SYNTAX

status **set_dft_isolation**
-ref_strategy *isolation_strategy*
-ref_domain *power_domain*
[-dft_target_domain *target_power_domain*]
[-type *type_list*]
[-exclude_type *exclude_type_list*]

Data Types

isolation_strategy string
power_domain string
target_power_domain *string*
type_list list
exclude_type_list list

ARGUMENTS

-ref_strategy *isolation_strategy*

Specify the strategy that will be applied to DFT ports. It shall be an existing element-based isolation strategy. If no **-type** is specified the strategy will apply to all DFT signal types in the domain. You can use **-type** for specific DFT signal types. Alternatively you can also use **-exclude_type** to exclude specific DFT signal types in the domain.

-ref_domain *power_domain*

Specify the domain where **-ref_strategy** is defined.

-dft_target_domain *target_power_domain*

Specify the target power domain of the connection.

-type *type_list*

Specify the connection types. Here is the list of supported connection types: scan_clock, scan_in, scan_out, scan_in_out, scan_enable, test_mode, wrp_clock, input_wrp_shift, output_wrp_shift, wrp_shift, wrp_ded_capture_in, wrp_ded_capture_out, wrp_ded_capture_inout, wrp_td_test, wrp_safe_in, wrp_safe_out, pse_clock, los_pipeline_enable, test_point_clock, test_point_test_mode.

-exclude_type *exclude_type_list*

Specify the connection types to be excluded. This option is mutually exclusive with **-type** option.

DESCRIPTION

This command specifies the strategy that will be applied to DFT ports punched by the tool.

EXAMPLES

The following example specifies isolation strategy iso1 on domain pd1 will be used for type scan_in.

prompt> **set_dft_isolation -ref_strategy iso1 -ref_domain pd1 -type scan_in**

SEE ALSO

set_isolation(2)
report_dft_isolation(2)

set_dft_location

Specifies the DFT hierarchy location for DFT insertion.

SYNTAX

status **set_dft_location**
dft_hier_name
[-include CODEC | WRAPPER | PLL | SERIAL_CNTRL]

Data Types

dft_hier_name string

ARGUMENTS

dft_hier_name

Specifies the hierarchical path name of an instance in the current design into which all DFT logic will be added during DFT insertion.

This argument is required.

-include CODEC | WRAPPER | PLL | SERIAL_CNTRL

Specifies the types of DFT logic to be included in the specified DFT location.

CODEC logic includes the compressor and decompressor (codec) logic inserted by the tool for compressed scan, serialized compressed scan, and streaming compressed scan modes.

PLL logic includes PLL controller and clock chain logic.

WRAPPER logic includes dedicated wrapper cells and wrapper mode logic.

SERIAL_CNTRL logic includes the serializer clock controller used in serializer insertion flows.

More than one type of DFT logic can be included with this option.

When this option is not specified, the location specification applies as a default to all DFT logic types.

DESCRIPTION

The **set_dft_location** command can be used to add all DFT logic into a user-specified hierarchy during DFT insertion.

If the **-include** option is not specified, the location specification applies as a default to all DFT logic types. Any type-specific specifications take precedence over this default location specification. You can use the **report_dft -location** command to report the resulting location specification.

In the normal serializer insertion flow, the deserializer and serializer registers are always placed inside the decompressor and compressor, respectively, whose location is specified with the CODEC logic type. To place the deserializer and serializer registers in a different hierarchy location than the combinational codec logic, use the serializer IP insertion flow and specify the register location with the SERIAL_REG logic type.

EXAMPLES

The following example specifies a DFT hierarchy location for the design top. All types of DFT logic are included in the specified location.

```
prompt> set_dft_location core_inst/dft_inst
Accepted DFT location specification.
1
```

The following example specifies a DFT hierarchy location for wrapper logic. The codec logic is inserted at another DFT location. Any other DFT logic should not be inserted in these locations.

```
prompt> set_dft_location core_inst/dft_inst1 -include { WRAPPER }
Accepted DFT location specification.
1
prompt> set_dft_location core_inst/dft_inst2 -include { CODEC }
Accepted DFT location specification.
1
```

SEE ALSO

remove_dft_location(2)
report_dft(2)

set_dft_signal

Defines the dft signal type for a set of signals.

SYNTAX

status **set_dft_signal**
-type *signal_type*
[-view *view_name*]
[-test_mode *mode_name_list*]
[-port *port_list*]
[-active_state 0 | 1
[-hookup_pin *hookup_pin*]
[-internal_clocks none | single | multi]
[-timing *timing*]
[-period *period*]
[-usage *use_type*]
[-ctrl_bits *ctrl_bits_list*]
[-pll_clock *clock_name*]
[-ate_clock *clock_name*]
[-connect_to *object_list*]
[-associated_internal_clocks *clock_pins_list*]

Data Types

signal_type string
view_name string
mode_name_list list
port_list list
hookup_pin list
timing list
period float
use_type list
ctrl_bits_list list
clock_name string
object_list list
clocks_pins_list list

ARGUMENTS

-view

Indicates the view to which the specification applies. The following views are valid:

- **existing_dft** implies that the specification refers to the existing usage of a port. For example, when working with a design

that is already DFT-inserted, the command to indicate that port SE is used as a scan-enable port is as follows:

set_dft_signal -view existing_dft -port SE -type ScanEnable

- **spec** (the default) implies that the specification refers to ports that the tool must use during DFT insertion. For example, when preparing a design for DFT insertion, the command to specify that port SE is used as a scan-enable port for DFT insertion is as follows:

set_dft_signal -view spec -port A -type ScanEnable

-test_mode ***mode_name_list***

Specifies the test mode(s) to which the specification applies. Test modes are created with the **define_test_mode** command. A value of **all** or **all_dft** causes the specification to apply to all test modes.

If the **-test_mode** option is not specified, the command applies to the current test mode, which is determined by the last **define_test_mode** or **current_test_mode** command executed. If no test modes have been defined, the command applies to the default test mode.

-type ***signal_type***

Specifies the signal type. The valid signal types are as follows:

- **Reset** is the asynchronous set or reset of the design.
- **Constant** is a continuously applied value to a port.
- **TestMode** is a continuously applied value to a test mode port that may have different values between test modes.
- **ScanDataIn** is one of the scan inputs of the design.
- **ScanDataOut** is one of the scan outputs of the design.
- **ScanEnable** is the shift enable of a MUX-D scan design.
- **ScanClock** is a clock that performs both scan shift and capture in a MUX-D scan design. (This type is equivalent to defining the clock as both the MasterClock and ScanMasterClock types; there is no CTL signal type called ScanClock.)
- **ScanMasterClock** is the clock responsible for scan shift. In an LSSD scan style, the ScanMasterClock is the A clock, and in a clocked scan style it is the shift clock.
- **MasterClock** is the clock responsible for capture.

In general, the MasterClock is used for referring to the capture clock and ScanMasterClock for referring to the shift clock. Use the ScanClock signal type, which simultaneously specifies both MasterClock and ScanMasterClock types.
If **-usage occ** is used, MasterClock specifies the output pin of the on-chip clock generator or the output pin of the user-instantiated on-chip clock controller. In both cases, the **-hookup_pin** option must be used. This is a constantly running clock signal that never turns off.

- **RefClock** is a constant running external ATE clock.

The following signal types can be used in IEEE 1149 synthesis and integration:

- **tdi** specifies the TDI port or bsd reg tdi access pin.
- **tdo** specifies the TDO port or bsd reg tdo access pin.
- **tdo_en** specifies the TDO port enable pin.
- **tck** specifies the TCK port.
- **tms** specifies the TMS port.

- **trst** specifies the TRST port.
- **CaptureEn** signal enables capture flop.
- **ShiftEn** signal enables shift flop.
- **UpdateEn** signal enables capture flop.
- **Select** specifies the scan-path selection signal, which selects Test Data Register(TDR) register.
- **bsd_shift_en** specifies the register access pin to be hooked up to TAP shift_dr pin when the instruction that selects the register is active.
- **bsd_capture_en** specifies the register access pin to be hooked up to TAP sync_capture_en pin when the instruction that selects the register is active. This signal is also used in core integration.
- **bsd_capture_dr** specifies the register access pin to be hooked up to TAP Capture-DR state on the negative edge of TCK when the instruction that selects the register is active.
- **bsd_update_en** specifies the register access pin to be hooked up to TAP sync_update_dr pin when the instruction that selects the register is active. This signal is also used in core integration.
- **bsd_update_dr** specifies the register access pin to be hooked up to TAP Update-DR state on the negative edge of TCK when the instruction that selects the register is active.
- **capture_clk** specifies the register access pin to be hooked up to TCK/clock_dr pin when the instruction that selects the register is active.
- **update_clk** specifies the register access pin to be hooked up to TCK/update_dr pin when the instruction that selects the register is active.
- **inst_enable** specifies the register access pin to be held active when the instruction that selects the register is active.
- **bist_enable** specifies the register access pin to be held active when the instruction that selects the register is active and TAP is in Run-Test-Idle state.
- **bist_clk** specifies the register access pin to be hooked up to TCK when the instruction that selects the register is active and TAP is in Shift-DR state. This signal is also used in core integration.
- **bsd_reset** specifies the register access pin to be hooked up to the Test-Logic-Reset FSM state signal when the instruction that selects the register is active. When the bsd_reset signal is not part of any instruction's test data register definition, it is hooked up directly to the Test-Logic-Reset state signal.
- **bsd_test_logic_reset** specifies the Test-Logic-Reset TAP FSM state signal.
- **bsd_run_test_idle** specifies the Run-Test-Idle TAP FSM state signal.
- **bsd_select_dr_scan** specifies the Select-DR TAP FSM state signal.
- **bsd_capture_dr** specifies the Capture-DR TAP FSM state signal.
- **bsd_shift_dr** specifies the Shift-DR TAP FSM state signal.
- **bsd_exit1_dr** specifies the Exit1-DR TAP FSM state signal.
- **bsd_pause_dr** specifies the Pause-DR TAP FSM state signal.
- **bsd_exit2_dr** specifies the Exit2-DR TAP FSM state signal.
- **bsd_update_dr** specifies the Update-DR TAP FSM state signal.
- **bsd_select_ir_scan** specifies the Select-IR TAP FSM state signal.

- **bsd_capture_ir** specifies the Capture-IR TAP FSM state signal.
- **bsd_shift_ir** specifies the Shift-IR TAP FSM state signal.
- **bsd_exit1_ir** specifies the Exit1-IR TAP FSM state signal.
- **bsd_pause_ir** specifies the Pause-IR TAP FSM state signal.
- **bsd_exit2_ir** specifies the Exit2-IR TAP FSM state signal.
- **bsd_update_ir** specifies the Update-IR TAP FSM state signal.
- **bsd_mode_in** is the mode signal for input BSR cells.
- **bsd_mode_out** is the mode signal for output BSR cells.
- **bsd_mode1_inout** is the mode signal for mode1 of BC_7 BSR cells.
- **bsd_mode2_inout** is the mode signal for mode2 of BC_7 BSR cells.

The following signal types can be used in IEEE 1687:

- **iTdi** Specifies Test Data input port for IEEE 1687 IP.
- **iTdo** Specifies Test Data output port for IEEE 1687 IP.
- **iTck** Specifies Test Clock port for IEEE 1687 IP.
- **iTrst** Specifies Test reset port for IEEE 1687 IP.
- **iCaptureEn** iCaptureEn signal enables capture flop.
- **iShiftEn** iShiftEn signal enables shift flop.
- **iUpdateEn** iUpdateEn signal enables update flop.
- **iSelect** Specifies the scan-path selection signal, which selects Test Data Register(TDR) register.

The following signal types can be used in IEEE 1838:

- **stapTdi** Specifies Test Data input port for STAP of the design.
- **stapTdo** Specifies Test Data output port for STAP of the design.
- **stapTms** Specifies Test Mode Select port for STAP of the design.
- **stapTck** Specifies Test Clock port for STAP of the design.
- **stapTrstn** Specifies Test reset port for STAP of the design.

-timing

Specifies the rise time and fall time for clocks. For example:

```
set_dft_signal -view existing_dft -type ScanClock \
  -port CLK -timing [list 45 55]
```

-period ***period***

Specifies the period of the on-chip clocking (OCC) reference clock. The period value is a floating point number in nanoseconds. When **-period** is used, the values specified after **-timing** indicate the leading edge and the trailing edge of the reference clock. The **-period** option is valid only on signals of the RefClock type.

-hookup_pin

Specifies the pin to which signals are to be connected. By default, the tool connects wires to the core side of identified signal ports, jumping pads and buffers as needed. The **-hookup_pin** argument overrides this behavior and instructs the tool to connect wires to the specified pin. The pin can be either a leaf pin or hierarchical pin. Validation ensures that the pin direction (driver or load) is consistent with the signal type. Verify that a specific access pin is associated with only one design port.

When the **-type** is a scan clock and **-view** is set to **existing_dft**, the **-hookup_pin** argument allows you to specify internal pins for the scan clocks along with their associated top-level clock port specified with **-port**. The resulting protocol is accurate and complete with real top-level clocks.

If hookup pins are specified, only one port can be specified in the *port_list*.

However, if the internal pins flow is enabled (**set_dft_drc_configuration -internal_pins**), a hookup pin can be specified without specifying a port. The hookup pin specified should be either on a black box or should have a proper global driver. If a proper global driver is not found, the tool will not stitch to the specified hookup pin.

-internal_clocks single | **none** | **multi**

Specifies the setting for an internal clock. An internal clock is defined as an internal signal driven by a multiplexer (or multiple input gate) output pin. This option applies only to the multiplexed flip-flop scan style and it is ignored for other scan styles.

Note that the output of clock gating cells that are introduced by the Synopsys Power Compiler will not be treated as internal clocks, even if the **-internal_clocks** option is enabled.

The **-internal_clocks** option can be set to the following values:

- **single** specifies that the tool treats any internal clocks in the design as separate clocks for the purpose of scan chain architecting. The **single** value instructs the tool to stop at the first buffer or inverter driving the flip-flops clock.
- **none** (the default) specifies that the tool does not treat internal clocks as separate clocks.
- **multi** specifies that the tool treats any internal clocks in the design as separate clocks for the purpose of scan chain architecting. The **multi** value instructs the tool to jump over buffers and inverters, stopping at the first multi-input gate driving the flip-flops clock.

-port

Indicates the list of ports on which to apply the specifications.

-active_state

Specifies the active states for the following signal types:

- **ScanEnable**
- **Reset**
- **Constant**
- **TestMode**
- **pll_reset**
- **pll_bypass**

The active state can be 0 or 1 and it specifies the active sense of the port (high or low) or an internal hookup only pin (no port is associated with the hookup pin). Wildcards and collections are supported.

-usage

Specifies the usage of the signal.

This option specifies that the tool is to use the specified signal for that purpose only.

The signal must be defined with suitable valid values when using it to connect design elements using the **set_dft_signal -connect_to** command. The valid values for *use_type* are as follows:

- **occ** specifies that the signal is an output pin of a clock generator or a free running clock. This usage is valid only for **-type MasterClock** and **-type TestMode**. For example:

  ```
  set_dft_signal -view spec -type TestMode -port TM_OCC -usage occ
  set_dft_signal -view spec -type MasterClock -hookup_pin U_pll/PLLOUT -usage occ
  ```

- **clock_gating** specifies that the signal is to be connected to the test pin of identified clock gating cells during DFT insertion. This usage is valid only for **-type ScanEnable** and **-type TestMode**.

- **scan** specifies that the signal is to be connected to the enable of scan elements during DFT insertion. This usage is valid only for **-type ScanEnable**.

Multiple usages can be specified for a signal by specifying them as a list. For example, a ScanEnable can be used to connect to test pins of clock gating cells as well as enabling scan elements.

For example,

```
set_dft_signal -view spec -port SE \
-type ScanEnable -usage {scan clock_gating}
```

-ctrl_bits ***ctrl_bits_list***

Lists triplets that specify the sequence of control bits needed to enable the propagation of the clock generator outputs. This option is used in the on-chip clocking (OCC) flow. The first element of each triplet is the cycle number (integer) indicating when the clock signal will be propagated. The second element is the pin name of the control bit (a valid design hierarchical pin name). The third element is the active value (0 or 1) of the control bit. For example:

```
set_dft_signal -type ScanEnable -hookup_pin pll_controller/clk \
 -pll_clock pll_i/pll_clk -ate_clock ate_clk \
 -ctrl_bits [list \
   0 clk_chain_i/clk_ctrl_data[0] 1 \
   1 clk_chain_i/clk_ctrl_data[1] 1 \
   2 clk_chain_i/clk_ctrl_data[2] 1] \
 -view existing_dft
```

-pll_clock ***clock_name***

Specifies the port name of the pll clock. This option is used in the on-chip clocking (OCC) flow.

-ate_clock ***clock_name***

Specifies the port name of the ATE clock. This option is used in the on-chip clocking (OCC) flow.

-connect_to ***object_list***

Specifies that the DFT signal should be connected from the port or hookup pin to only the specified design objects.

You can use the **-connect_to** option to define ScanEnable, TestMode, wrp_shift or LOSPipelineEnable signals that should only be connected to a specific set of design objects. These are known as object-specific signal definitions. Depending on the signal type, only certain design object types can be specified and certain signal usages are supported. For the different signal types and usages defined with the **-type** and **-usage** options, the allowed object types for the **-connect_to** option are:

- **-type ScanDataIn**
  - **pin** (ScanDataIn pins of CTL-modeled cores)
- **-type ScanDataOut**

    - **pin** (ScanDataOut pins of CTL-modeled cores)
- **-type ScanEnable -usage scan**
    - **cell** (leaf scan cells)
    - **cell** (hierarchical cell containing scan cells)
    - **design** (designs containing scan cells)
    - **clock** (scan cells clocked by specified clocks)
    - **pin** (ScanEnable pins of CTL-modeled cores)
- **-type ScanEnable | TestMode -usage clock_gating**
    - **cell** (clock-gating cells)
    - **cell** (hierarchical cell containing clock-gating cells)
    - **design** (designs containing clock-gating cells)
    - **pin** (ScanEnable or TestMode pins of CTL-modeled cores)
- **-type wrp_shift | input_wrp_shift | output_wrp_shift**
    - **cell** (leaf wrapper cells) - highest priority
    - **pin** (specified wrp_shift pins of CTL-modeled cores)
    - **cell** (wrp_shift pins for all wrapper segments in CTL-modeled cores)
    - **port** (wrapper cells associated with specified ports)
    - **clock** (wrapper cells clocked by specified clocks) - lowest priority
- **-type LOSPipelineEnable**
    - **clock** (pipeline scan enable logic clocked by specified clocks)

For the **clock_gating** usage, you can specify ScanEnable or TestMode signal definitions, depending on the type of clock-gating control signal (scan-enable or test-mode) used by the specified objects.

Pins of CTL-modeled cores are not supported when performing simple HSS integration of standard scan cores.

When you specify a clock-domain-based signal specification by specifying clocks, they must be defined as valid test clocks with the **set_dft_signal** command. The propagation of the test clock determines the scope of the specification; the propagation of any underlying functional clocks at the clock source is not considered.

When using clock-domain-based signal specifications with CTL-modeled cores, the following requirements must be observed during core creation to ensure that the CTL models contain clock-domain information:

- Core-level ScanEnable signals must be defined using the **-usage** option of the **set_dft_signal** command. This ensures that the core's internal scan-enable signal is not used outside its intended usage.
- Core-level ScanEnable signals defined with a usage of **clock_gating** must also be defined as domain-specific signals using the **-connect_to** *clock_list* option of the **set_dft_signal** command. This ensures that clock-specific clock-gating annotations are included in the CTL model.

If the cores do not contain clock-domain information, you can directly specify the ScanEnable pins of the CTL-modeled core instead.

The following example defines two clock-domain-based ScanEnable signals:

```
set_dft_signal -view exist -type ScanClock -timing {45 55} \
  -port {CLK1 CLK2}

set_dft_signal -type ScanEnable -port SE1 -usage scan \
  -connect_to CLK1
set_dft_signal -type ScanEnable -port SE2 -usage scan \
  -connect_to CLK2
```

The following example defines a ScanEnable signal to be used only as the clock-gating control signal for a specific block:

```
set_dft_signal -type ScanEnable -port SE_CG_IP -usage clock_gating \
  -connect_to UIP_CORE
```

The **-connect_to** option is also used in the hierarchical on-chip clocking (OCC) flow to specify the correspondence between the top-level ATE clock port and the core-level ATE clock pins. For existing top-level connections, use the **-view existing** option. For connections to be made by DFT insertion, use the **-view spec** option. For example,

```
set_dft_signal -view spec -port TOP_CLK -type MasterClock \
  -connect_to {CORE1/ATECLK CORE2/ATECLK} -timing {45 55}
```

Clock-domain-based wrapper shift signals are supported.

-associated_internal_clocks ***clock_pins_list***

Specifies a list of internal pins associated to be associated with a clock source port.

The tool assumes that the port-driven clock signal is also driven at these internal pins, with the same timing waveform and polarity. The fanout from each internal clock pin is treated as a separate clock domain when architecting scan chains, and the pin names are shown as the clock names by the **preview_dft** command.

Only leaf pins can be specified; hierarchical pins are not supported.

For example,

```
set_dft_signal -type ScanClock -port CLK \
 -associated_internal_clocks [list U1/int_clk U2/int_clk1 U1/int_clk2] \
 -view existing
```

During pre-DFT DRC, the tool does not check for logical connectivity between the port and internal clock pins. This provides a method to bypass black boxes or other logic with unknown functionality in the clock network.

The connectivity is checked in post-DFT DRC.

DESCRIPTION

The **set_dft_signal** command can be used to specify one or more primary input or output ports as DFT signals. It can be used either to describe the existing usage or to prescribe the usage during implementation.

A DFT signal has a port and a signal type. You can specify multiple DFT signals, but you can specify only one signal type. Port directions must be consistent with the signal type.

You can use the **set_dft_signal** command to declare different DFT signals for different test modes. For example, each test mode can have different scan-in and scan-out ports, or different constant, set, or reset signals. However, all test modes share the same scan-enable signals. You cannot specify different scan-enable signals for different test modes.

The **set_dft_signal** commands are not incremental. A **set_dft_signal** command that identifies a port overwrites any previous **set_dft_signal** command that identifies the same port.

To review your DFT signal specifications, use the **report_dft** command.

Descriptive specification (using **-view existing_dft**) will invalidate test protocol and details about DFT structures.

EXAMPLES

The following example defines port myStapTdi for stapTdi signal.

```
prompt> set_dft_signal -type stapTdi -port myStapTdi
Information: Accepted stapTdi dft signal specification in partition 'default_partition' for all_dft mode (DFT-1405)
1
```

SEE ALSO

preview_dft(2)
report_dft(2)
set_dft_drc_configuration(2)
create_test_protocol(2)
dft_drc(2)

set_disable_auto_mux_clock_exclusivity

Disables automatic inference of MUXed clock exclusivity at specified cell output pins.

SYNTAX

int **set_disable_auto_mux_clock_exclusivity**
[object_list]

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of one or more MUX output pins for which automatic inference of MUXed clock exclusivity is disabled.

DESCRIPTION

Automatic inference of clock exclusivity at MUX output pins is enabled by setting the time.enable_auto_mux_clock_exclusivity app option to true.

To prevent automatic inference of a specific MUX cell output as an exclusivity point, apply this command to the MUX output pin.

To undo the set_disable_auto_mux_clock_exclusivity command, use the remove_clock_exclusivity command on the same output pin.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example prevents pins MUX02/Z and MUX04/Z from being automatically inferred as clock exclusivity points.

```
prompt> set_disable_auto_mux_clock_exclusivity {MUX02/Z MUX04/Z}
```

SEE ALSO

set_clock_exclusivity(2)
remove_clock_exclusivity(2)
report_clock(2)
time.enable_auto_mux_clock_exclusivity(3)

set_disable_clock_gating_check

Disables the clock gating check for specified cells and pins in the current design.

SYNTAX

string **set_disable_clock_gating_check**
object_list

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of cells and pins for which the clock gating check is disabled.

DESCRIPTION

The **set_disable_clock_gating_check** command disables clock gating checks that are inferred automatically and possibly modified by the **set_clock_gating_check** command. It does not affect clock gating checks defined in the library for integrated clock-gating cells.

You can disable clock gating checks on cells or pins in the current design or its subdesigns. Specify cells at lower levels in the design hierarchy as *instance1*/*instance2*/*cell_name*. Specify pins at lower levels in the design hierarchy as *instance1*/*instance2*/*cell_name*/*pin_name*. For subdesigns in the hierarchy, specify instance names, not design names.

Specifying a cell disables all gating checks in the cell. Specifying a pin disables any gating check in which the pin is the gating clock pin or gating enable pin.

To restore gating checks previously disabled by the **set_disable_clock_gating_check**, use the **remove_disable_clock_gating_check** command. To globally disable all clock gating checks, including clock gating checks defined in the library, set the **time.disable_clock_gating_checks** application option to **true**.

EXAMPLES

The following command disables all clock gating checks on cell U123.

prompt> **set_disable_clock_gating_check [get_cells U123]**

The following command disables clock gating checks that use pin U123/I as a gating enable pin or gating clock pin.

prompt> **set_disable_clock_gating_check [get_pins U123/I]**

SEE ALSO

set_clock_gating_check(2)
remove_clock_gating_check(2)
remove_disable_clock_gating_check(2)
report_clock_gating_check(2)
time.disable_clock_gating_checks(3)

set_disable_tie_insert

To exclude some pins for the insertion of tie cell.

SYNTAX

status **set_disable_tie_insert**
[-objects *object_name_or_collection*]

Data Types

object_name_or_collection collection

ARGUMENT

-objects *object_name_or_collection*

Specifies the objects for exclusion during tie cell insertion. Currently supported object types are pins and ports.

DESCRIPTION

set_disable_tie_insert command marks the pins on which tie cell will not be inserted by add_tie_cells and optimization commands.

EXAMPLES

prompt>set_disable_tie_insert -objects [get_pins {h1/reg_2_/SE h1/reg_2_/SI}]

prompt>set_disable_tie_insert -object h1/reg_2_/SE

SEE ALSO

add_tie_cells(2)
reset_disable_tie_insert(2)
report_disable_tie_insert(2)

set_disable_timing

Disables timing arcs in a circuit.

SYNTAX

string **set_disable_timing**
[-from *from_pin_name* -to *to_pin_name*]
object_list

string *from_pin_name*
string *to_pin_name*
list *object_list*

ARGUMENTS

-from *from_pin_name*

Specifies that arcs only from this pin on the specified cell are to be disabled. The *from_pin_name* value must be a valid library pin name corresponding to the cell in *object_list*. When used, the **-from** and **-to** options must be specified together. The *object_list* must contain only cells, and the command specifies that arcs only between these two pins on the specified cells are to be disabled.

-to *to_pin_name*

Specifies that arcs only to this pin on the specified cell are to be disabled. The *to_pin_name* value must be a valid library pin name corresponding to the cell in *object_list*. When used, the **-from** and **-to** options must be specified together. The *object_list* must contain only cells, and the command specifies that arcs only between these two pins on the specified cells are to be disabled.

object_list

Specifies a list of cells, pins, lib-cells, lib-pins, ports, or timing-arcs to be disabled. This list can include library objects.

DESCRIPTION

Disables timing through the specified cells, pins, ports, or timing arcs in the **current_mode**, if library cells or pins are specified, it disables the timing arcs for all modes that utilize the library.

Any cell, pin, port, or timing arc in the **current_design** or its subdesigns can be disabled. Cells at lower levels in the design hierarchy are specified as *instance1*/*instance2*/*cell_name*. Pins at lower levels in the design hierarchy are specified as *instance1*/*instance2*/*cell_name*/*pin_name*. Ports at lower levels in the design hierarchy are specified as *instance1*/*instance2*/*cell_name*/*port_name*. Timing arcs have to be collected using the *get_timing_arcs* command.

Note: For subdesigns in the hierarchy, you must specify instance names instead of design names.

When the timing through a cell is disabled, only those cell arcs are disabled that lead to the output pins of the cell. When the timing through a pin or port is disabled, only those cell arcs that lead from a load pin and to a driver pin are disabled. For bidirectional pins or ports the cell arcs from the load side and to the driver side are disabled.

When **-from** and **-to** pins are specified, all arcs between these two pins on the cell are disabled.

To view disabled timing arcs in the current design, use the **report_disable_timing** command. To restore timing arcs disabled by **set_disable_timing**, use the **remove_disable_timing** command. The **reset_design** command removes all user-specified attributes from the current design, including those set by **set_disable_timing**.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

prompt> **set_disable_timing -from A -to Z [get_cells { CO }]**

prompt> **set_disable_timing [get_timing_arcs -from { CO/A } -to { CO/Z }]**

SEE ALSO

remove_disable_timing(2)
report_timing(2)
reset_design(2)
get_timing_arcs(2)

set_domain_supply_net

Sets the primary power net and primary ground net of an existing power_domain.

SYNTAX

int **set_domain_supply_net**
-primary_power_net *supply_net_name*
-primary_ground_net *supply_net_name*
domain_name

Data Types

supply_net_name string
domain_name string

ARGUMENTS

domain_name

Specifies the name of the power domain. If a power domain with the specified name does not exist, in the current scope, this command fails.

The domain_name option is a required option and must be specified.

-primary_power_net *supply_net_name*

Specifies the power net of the power domain. If a supply net with the specified name does not exist in the current scope, the command fails. If the supply net is not associated with specified power domain, the command fails.

This is a required option and must be specified.

-primary_ground_net *supply_net_name*

Specifies the ground net of the power domain. If a ground net with the specified name does not exist in the current scope, the command fails. If the ground net is not associated with specified power domain, the command fails.

This is a required option and must be specified.

DESCRIPTION

The *set_domain_supply_net* command enables sets the primary power net and the primary ground net of a power domain. All extent of the power domain shares the same power/ground supply until explicitly excluded. Here "explicitly excluded" means it is

explicitly connected with another supply_net (non primary power/ground supply_net) by the **connect_supply_net** command. This command errors out if power domain already has a primary power and ground net.

The supply_net must be created in the same power_domain at first before it could be set as the primary power or ground supply_net. Use the **create_supply_net** command to create a supply_net at a specified power_domain.

This command cannot be used to specify the supply nets of a power domain which was created with **create_power_domain -supply {primary ..}**.

Command returns 1 on success, returns 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates two supply_net on power_domain PD1, then set them as primary power or ground net.

```
prompt> create_power_domain PD1 -elements INST_1
PD1
prompt> create_supply_net A_VDD -domain PD1
A_VDD
prompt> create_supply_net A_VSS -domain PD1
A_VSS
prompt> set_domain_supply_net PD1 -primary_power_net A_VDD \
  -primary_ground_net A_VSS
1
```

SEE ALSO

connect_supply_net(2)
create_power_domain(2)
create_supply_net(2)

set_dont_retime

Prevents retiming optimization from moving registers. Command can also be used to selectively re-enable retiming on registers.

SYNTAX

int **set_dont_retime**
object_list
[true | false]

Data Types

object_list collection

ARGUMENTS

object_list

Specifies collection of registers, hierarchical cells, modules or design. If hierarchical cells are specified, all registers inside hierarchical cell are excluded from (or included into) retiming. If modules are specified, all registers inside all instances of modules are excluded from (or included into) retiming. If design is specified, all registers inside design are excluded from (or included into) retiming.

true | **false**

True excludes objects from retiming. False includes registers into retiming. Default is true.

DESCRIPTION

set_dont_retime set to true excludes registers from retiming optimization. **set_dont_retime** set to false includes registers into retiming optimization.

Applying **set_dont_retime** to true(false) on a hierarchical cell prevents(allows) retiming of all registers inside hierarchical cell. Applying **set_dont_retime** to true(false) on a leaf register prevents(allows) retiming of that register. Applying **set_dont_retime** to true(false) on a module is same as applying **set_dont_retime** to true(false) on all instances of modules. Applying **set_dont_retime** true(false) to design prevents retiming (allows retiming) of all registers in the design.

Applying **set_dont_retime** on a hierarchical object (module or hierarchical instance) takes effect on all lower, nested hierarchies of objects.

You can apply **set_dont_retime** to false if you want to selectively allows retiming on the object whose parent object has **set_dont_retime** set to true. Example below illustrates one such use.

Please note, allowing register to retime does not necessarily mean that it will get retimed. Retiming optimization will make decision based on improvement to area or delay cost.

EXAMPLES

The following example excludes cells named *z1_reg* and *z2_reg* from retiming.

```
prompt> set_dont_retime [get_cells {z1_reg z2_reg}] true
```

The following example specifies that the cell named *H1/z_reg* is allowed to retime, but other sequential cells inside *H1* are not allowed into retiming.

```
prompt> set_dont_retime [get_cells "H1"] true
prompt> set_dont_retime [get_cells "H1/z_reg"] false
```

The following example specifies that no sequential cells inside any instance of module *Controller* can be moved by retiming.

```
prompt> set_dont_retime [get_modules "Controller"] true
```

SEE ALSO

compile(2)
set_optimize_registers(2)

set_dont_touch

Sets the **dont_touch** attribute on cells, nets, designs, and library cells to prevent the tool from replacing or modifying them during optimization.

SYNTAX

status **set_dont_touch**
[*object_list*]
[*value*]

Data Types

object_list collection
value Boolean

ARGUMENTS

object_list

Specifies the objects on which to set the **dont_touch** attribute. You can set the attribute on cells, nets, designs, and library cells.

value

Specifies the value of the **dont_touch** attribute. Valid values are true (the default) or false.

DESCRIPTION

This command sets the **dont_touch** attribute on cells and nets in the current design or on designs and library cells. The **dont_touch** attribute prevents modification or replacement of these objects during optimization.

Setting the **dont_touch** attribute on a hierarchical cell implies a **dont_touch** attribute on all cells below it.

Setting the **dont_touch** attribute on a design affects the design only when the design is instantiated within another design as a level of hierarchy. In this case, the **dont_touch** attribute on the design implies that all cells under that level of hierarchy have the **dont_touch** attribute. Setting the **dont_touch** attribute on the top-level design has no effect because it is not instantiated within any other design. If the **dont_touch** attribute is on a design that is removed, the attribute continues to apply to any cells that were instances of that design until the next time the current design is linked.

Setting the **dont_touch** attribute on a library cell implies a **dont_touch** attribute on all instances of that cell.

Note that the **dont_touch** attribute applied by the tool on the cells and nets in the transitive fanout of objects with the **dont_touch_network** attribute overrides the **dont_touch** attribute setting of false that is set by the **set_dont_touch** command. To

report the **dont_touch** value on cells or nets, use the **report_cell** or **report_net** command.

To remove the **dont_touch** attribute, use the **remove_attributes** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following commands prevent modification or removal of the block1 and analog1 cells during optimization but allow the N1 net to be modified.

```
prompt> set_dont_touch [get_cells {block1 analog1}]
1
prompt> set_dont_touch  [get_nets N1] false
1
```

The following example shows that the instances of a design inherit the **dont_touch** attribute setting from the design.

```
prompt> get_attribute [get_cells u1] ref_name
d1
prompt> get_attribute [get_cells u1] dont_touch
false
prompt> set_dont_touch [get_designs d1]
1
prompt> get_attribute [get_cells u1] dont_touch
true
```

SEE ALSO

get_attribute(2)
report_dont_touch(2)
set_dont_touch_network(2)

set_dont_touch_network

Sets **dont_touch_network** on clocks, pins, or ports in the current design to prevent cells and nets in the transitive fanout of the **set_dont_touch_network** objects from being modified or replaced during optimization.

SYNTAX

status **set_dont_touch_network**
[-no_propagate]
[-clock_only]
[-clear]
object_list

Data Types

object_list collection

ARGUMENTS

-no_propagate

Indicates that the dont_touch network is not propagated through logic gates. This option is ignored for dont_touch_network of clocks.

-clock_only

Indicates that the dont_touch network is propagated only through clock networks. This option is redundant for dont_touch_network of clocks.

-clear

Indicates that the pins, ports, or clocks should no longer be sources of dont_touch_network.

object_list

Specifies the pins, ports, or clocks in the current design on which to spread network dont_touch.

DESCRIPTION

This command sets the clocks, pins, or ports in the current design as source of network dont_touch. The tool will assign **dont_touch** attributes to all cells and nets in the transitive fanout of these objects so that they are not modified or replaced during optimization. A **dont_touch** attribute assigned by the tool using the **set_dont_touch_network** command overrides the **false dont_touch** attribute value from the **set_dont_touch** command.

If you specify a pin or a port, all nets and cells in its transitive fanout are recursively marked **dont_touch**. This recursion propagates through combinational cells but stops at registers (without marking the registers as **dont_touch**). An element is recognized as a register only if it is a timing endpoint. If you specify an input pin, its own cell is marked **dont_touch** but its connected net is not marked **dont_touch**. If you specify an output pin, its own cell is not marked **dont_touch** but its connected net is marked **dont_touch**.

If you specify a clock, the transitive fanout of all the clock's sources is affected. The **set_dont_touch_network** command is intended primarily for clock circuitry. Placing a **dont_touch_network** on a clock object prevents the tool from modifying the clock buffer network during optimization. The **-clock_only** and **-no_propagate** options will be ignored for clocks.

The **set_dont_touch_network** command does not prevent clock gating optimizations from modifying or removing clock gating cells. To prevent this, use the **set_dont_touch** command instead.

If you specify the **-no_propagate** option, the dont_touch network is not propagated through cell arcs.

To see the **dont_touch** value on the cells or nets, use the **report_dont_touch**, **report_cells** or **report_nets** commands. The **report_dont_touch** and **report_nets** commands display the nets that are implicitly **dont_touch** as a result of using the **set_dont_touch_network** command.

To remove **dont_touch_network** from a pin, port, or clock, use the **-clear** option with **set_dont_touch_network** command, on the specific object on which **dont_touch_network** had been set.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following command sets **dont_touch_network** on a port named clock_in:

```
prompt> set_dont_touch_network clock_in
```

The following command clears **dont_touch_network** from the port named clock_in:

```
prompt> set_dont_touch_network -clear [get_port clock_in]
```

The following command sets **dont_touch_network** on a net named net1 and a cell named cell2 but not to the cell named cell1:

```
prompt> get_pins -of_objects [get_nets net1]
{cell1/Y cell2/A}
prompt> set_dont_touch_network [get_pins cell1/Y]
1
```

The following command sets **dont_touch_network** on a net named net1 and cells named cell1 and cell2 but not on the net named net0:

```
prompt> get_pins -of_objects [get_nets net1]
{cell1/Y cell2/A}
prompt> get_pins -of_objects [get_cells cell1]
{cell1/A cell1/Y}
prompt> get_nets -of_objects [get_pins cell1/A]
{net0}
prompt> set_dont_touch_network [get_pins cell1/A]
1
```

SEE ALSO

report_cells(2)
report_clocks(2)
report_design(2)
report_dont_touch(2)
report_nets(2)
report_ports(2)
set_dont_touch(2)

set_drive

Sets the resistance to a specified value on specified input or inout ports in the current design.

SYNTAX

string **set_drive**
  [-rise] [-fall]
  [-min] [-max]
  [-modes *mode_list*]
  [-corners *corner_list*]
  [-scenarios *scenario_list*]
  resistance port_list

list *mode_list*
list *corner_list*
list *scenario_list*
float *resistance_value*
list *port_list*

ARGUMENTS

-rise

Indicates that *resistance_value* is to be used to drive the ports only for the rising case.

-fall

Indicates that *resistance_value* is to be used to drive the ports only for the falling case.

-min

Applies only to designs in min-max mode (min and max operating conditions). Indicates that the *resistance_value* is the minimum resistance.

-max

Applies only to designs in min-max mode (min and max operating conditions). Indicates that the *resistance_value* is the maximum resistance.

-modes *mode_list*

Specifies the scenarios that the resistance value will be applied to. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option may not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by -corners will be used.

-corners *corner_list*

Specifies the scenarios that the resistance value will be applied to. If this option is given, all scenarios of the given corners and the current mode will be used. The **-scenarios** option may not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by -corners will be used.

-scenarios ***scenario_list***

Specifies the scenarios that the resistance value will be applied to. The **-modes** or **-corners** option may not be given with this option.

resistance

Specifies a nonnegative port drive resistance value for the ports in *port_list*. The value must be >= 0; units must be the same as those in the technology library.

port_list

Specifies a list of input or inout ports in the current design, on which the *resistance_value* is to be set.

DESCRIPTION

Sets the resistance to a specified value on specified input or inout ports in the current design. The tool models the driver as a resistance and uses that value to calculate the wire delay of the port. To view the drive that is set, use **report_port -drive**.

Depending on the options used, **set_drive** sets these attributes on the specified ports:

drive_resistance_fall_max
drive_resistance_fall_min
drive_resistance_rise_max
drive_resistance_rise_min

If **set_drive** is issued without any options, the **drive_resistance_fall_max** and **drive_resistance_rise_max** attributes are set; in min-max mode, the other two attributes are also set.

If **set_drive** is issued with only the **-rise** option, only the **drive_resistance_rise_max** attribute is set; in min-max mode, **drive_resistance_rise_min** is also set.

If **set_drive** is issued with only the **-fall** option, only the **drive_resistance_fall_max** attribute is set; in min-max mode, **drive_resistance_fall_min** is also set.

Drive resistance is managed on a per-scenario basis. For designs with multiple scenarios, you can specify different resistances for different scenarios by using **-modes**, **-corners**, and **-scenarios** options. By default, if none of these options are specified, the given resistance is applied to the current scenario. (It will be an error if there is no current scenario). If the **-scenarios** option is given, the resistance will be applied to all of the specified scenarios. If the **-modes** option is given, the resistance will be applied to all scenarios of the specified modes. (It will be an error if none of the specified modes have any scenarios.) If the **-corners** option is given, the resistance will be applied to all scenarios of the specified corners and the current mode. (It will be an error if none of the specified corners have any scenarios with the current mode.) If both the **-modes** and **-corners** options are given, the resistance will be applied to all scenarios of the specified corners and the specified modes. (It will be an error there are no such scenarios.) It is an error to give -scenarios with **-modes** or **-corners**. Drive resistance is the output resistance of the cell that drives the port, so that a higher drive resistance means less drive capability and longer delays. Thus, a drive *resistance_value* of 0 is infinite drive, or no delay between the ports and all ports connected to them. Setting *resistance_value* to 0 is the same as removing the drive resistance with **remove_drive_resistance**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example sets the drive resistance to 5 on all input ports in the current scenario of the current design, and displays the ports on which the drive resistance has been set.

```
prompt> set_drive 5 [all_inputs]
1
prompt> report_port -drive
****************************************
Report : port
       -drive
Design : counter
****************************************

                Resistance          Transition
Input Port    Rise      Fall      Rise      Fall
------------------------------------------------------------
A             5.00      5.00      --        --
B             5.00      5.00      --        --
C             5.00      5.00      --        --
CL            5.00      5.00      --        --
CLK           5.00      5.00      --        --
D             5.00      5.00      --        --
JOKE          5.00      5.00      --        --
L             5.00      5.00      --        --
P             5.00      5.00      --        --
RESET         5.00      5.00      --        --
T             5.00      5.00      --        --
```

SEE ALSO

remove_drive_resistance(2)
report_port(2)
set_load(2)

set_drive_resistance

Sets drive resistance for input or inout ports.

Note: This command is obsolete, and has been replaced by the **set_drive** command. Use **set_drive** instead.

SYNTAX

string **set_drive_resistance** [-rise]
[-fall]
[-min]
[-max]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
resistance
port_list

float *resistance*
list *port_list*
list *mode_list*
list *corner_list*
list *scenario_list*

ARGUMENTS

-rise

Specifies to use the resistance to drive the ports for the rising case.

-fall

Specifies to use the resistance to drive the ports for the falling case.

-min

Specifies to use the resistance to drive the ports for the minimum case.

-max

Specifies to use the resistance to drive the ports for the maximum case.

-modes *mode_list*

Specifies the scenarios that the resistance value will be applied to. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option may not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode

is specified by **-modes** and whose corner is specified by -corners will be used.

-corners ***corner_list***

Specifies the scenarios that the resistance value will be applied to. If this option is given, all scenarios of the given corners and the current mode will be used. The **-scenarios** option may not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by -corners will be used.

-scenarios ***scenario_list***

Specifies the scenarios that the resistance value will be applied to. The **-modes** or **-corners** option may not be given with this option.

resistance

Specifies a nonnegative resistance value for the ports. Port drive resistance (value >= 0).

port_list

Specifies a list of input or inout port names of the current design on which the drive attributes are to be set.

DESCRIPTION

Model the driver as a resistance and use that value to calculate the wire delay of the port. **report_port -drive** can be used to view the drive which is set.

Drive resistance is the output resistance of the cell that drives the port; such that a higher drive resistance means less drive capability and longer delays. Thus, a drive *resistance* of 0 is infinite drive, or no delay between the ports and all connected to them. This is the same as removing the drive resistance with **remove_drive_resistance**.

Drive resistance must be in units with the technology library.

Drive resistance is managed on a per-scenario basis. For designs with multiple scenarios, you can specify different resistances for different scenarios by using **-modes**, **-corners**, and **-scenarios** options. By default, if none of these options are specified, the given resistance is applied to the current scenario. (It will be an error if there is no current scenario). If the **-scenarios** option is given, the resistance will be applied to all of the specified scenarios. If the **-modes** option is given, the resistance will be applied to all scenarios of the specified modes. (It will be an error if none of the specified modes have any scenarios.) If the **-corners** option is given, the resistance will be applied to all scenarios of the specified corners and the current mode. (It will be an error if none of the specified corners have any scenarios with the current mode.) If both the **-modes** and **-corners** options are given, the resistance will be applied to all scenarios of the specified corners and the specified modes. (It will be an error there are no such scenarios.) It is an error to give -scenarios with **-modes** or **-corners**.

Multicorner-Multimode Support

This command works on the current scenario.

EXAMPLES

```
prompt> set_drive_resistance 5 [all_inputs]
1
prompt> report_port -drive
****************************************
```

```
Report : port
       -drive
Design : counter
****************************************

                Resistance         Transition
Input Port    Rise      Fall      Rise      Fall
-----------------------------------------------------------------
A             5.00      5.00      --        --
B             5.00      5.00      --        --
C             5.00      5.00      --        --
CL            5.00      5.00      --        --
CLK           5.00      5.00      --        --
D             5.00      5.00      --        --
JOKE          5.00      5.00      --        --
L             5.00      5.00      --        --
P             5.00      5.00      --        --
RESET         5.00      5.00      --        --
T             5.00      5.00      --        --
```

SEE ALSO

remove_drive_resistance(2)
report_port(2)
set_drive(2)
set_load(2)

set_driving_cell

Sets the port driving cell.

SYNTAX

string **set_driving_cell**
-lib_cell *lib_cell_name*
[-rise]
[-fall]
[-min]
[-max]
[-library *lib_name*]
[-pin *pin_name*]
[-from_pin *from_pin_name*]
[-multiply_by *factor*]
[-dont_scale]
[-no_design_rule]
[-input_transition_rise *rising_transition*]
[-input_transition_fall *falling_transition*]
[-clock *clock_name*]
[-clock_fall]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
port_list

Data Types

lib_cell_name string
lib_name string
pin_name string
from_pin_name string
factor float
rising_transition float
falling_transition float
clock_name string
mode_list list
corner_list list
scenario_list list
port_list list

ARGUMENTS

-lib_cell *lib_cell_name*

Specifies the name of the library cell used to drive the ports. You must use the **-pin** option if the cell has more than one output pin. If different cells are needed for the rising and the falling cases, use separate commands with the **-rise** or **-fall** option. Use the **-from_pin** option to choose between multiple input pins with arcs to this output pin. This option is required.

-rise

Sets driving cell information for only the rising port transition.

-fall

Sets driving cell information for only the falling port transition.

-min

Sets driving_cell information for only the minimum analysis. This option is valid only in min-max mode.

-max

Sets driving cell information for only the maximum analysis. This option is valid even when not in min-max mode. When not in min-max mode, the option is not required because it is the default.

-library *lib_name*

Specifies the name of the library containing the *library_cell_name* value. This is the library of the driving cell. By default, the libraries in the search path are searched for the cell.

-pin *pin_name*

Uses the driver for the specified output pin. This is the driving pin name. If you do not include the **-from_pin** option, the command uses an arbitrary pin arc ending at the specified pin.

-from_pin *from_pin_name*

Specifies an input pin on the specified cell so the command uses the driver of the timing arc from this pin to the specified pin.

-multiply_by *factor*

Multiplies the calculated transition time by the specified multiplier. *factor* must be greater than or equal to 0.

-dont_scale

Prevents operating condition scaling. When you specify this option, the timing analyzer does not scale the drive capability of the ports according to the current operating conditions. By default, the port drive capability is scaled for operating conditions exactly as the driving cell itself would have been scaled.

-no_design_rule

Skips the application of design rules from the driving cell to the port. If you do not include this option, the design rules (such as max_capacitance) of the library pin are applied to the port.

-input_transition_rise *rising_transition*

Specifies the input rising transition time associated with the **-from_pin** option. Use the **-input_transition_rise** and **-input_transition_fall** options to capture the accurate transition time associated with the *from_pin_name* value. By using this option, you can obtain more accurate information on the transition time and delay time at the output pin. If you do not specify this option, the default is 0.

-input_transition_fall *falling_transition*

Specifies the input falling transition time associated with the *from_pin_name* value. If you do not specify this option, the default value is 0.

-clock ***clock_name***

Sets the driving cell relative the specified clock.

-clock_fall

Specifies that driving cell is relative to the falling edge of the clock. The default is the rising edge.

-modes ***mode_list***

Specifies the scenarios to use for the driving cell. If this option is given, all scenarios of the specified modes will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners ***corner_list***

Specifies the scenarios to use for the driving cell. If this option is given, all scenarios of the specified corners and the current mode will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios ***scenario_list***

Specifies the scenarios to use for the driving cell. The **-modes** or **-corners** option must not be specified with this option.

port_list

Provides a list of input ports. The list contains input or inout port names in the current design on which the driving cell information is set.

DESCRIPTION

Sets the port driving cell, and sets attributes on the specified input or inout ports in the current design to associate an external driving cell with the ports. The drive capability of the port is the same as if the specified driving cell were connected in the same context to allow accurate modeling of port drive capability for nonlinear delay models. If you omit the **-dont_scale** option, the drive capability of the port is scaled according to the current operating conditions. Use the **report_ports** command with the **-drive** option to view drive information on ports.

The driving cell can be relative to a clock by using **-clock** option. This means the driving cell apply only to those external paths driven by the clock. Using **-clock** or **-clock_fall** option can be meaningful only when timing_slew_propagation_mode is in worst_arrival mode. Note if it is in worst_slew mode, there must be a driving cell without any clock specified to see any driving cell on the port. If a clock-relative driving cell is set on a port, these driving-cell attributes will not be accessible via **get_attribute** until the **-clock** and **-clock_fall** options are supported for **set_driving_cell** in SDC.

Driving cell data is managed on a per-scenario basis. For designs with multiple scenarios, you can specify different driving cell settings for different scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command applies to the current scenario. The tool issues an error if there is no current scenario. If the **-scenarios** option is given, the command applies to all of the specified scenarios.

If the **-modes** option is given, the command applies to all scenarios of the specified modes. The tool issues an error if none of the specified modes have any scenarios.

If the **-corners** option is given, the command applies to all scenarios of the specified corners and the current mode. The tool issues an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are given, the command applies to all scenarios of the specified corners and the specified modes. The tool issues an error if there are no such scenarios. It is an error to specify **-scenarios** with **-modes** or **-corners**.

Use the **remove_driving_cell** command to remove driving cell information from ports.

Note: The **set_driving_cell** command removes any corresponding drive resistance or input transition attributes (from the **set_drive** command or the **set_input_transition** command) on the specified ports. If possible, always use the **set_driving_cell** command instead of the **set_drive** command, because **set_driving_cell** allows accurate calculation of port delay and transition time for library cells with nonlinear dependence on capacitance.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example sets port A of the AND cell as the driving cell.

```
prompt> set_driving_cell \
  -lib_cell AND [get_ports A]

prompt> report_ports -drive A
****************************************
Report : port
       -drive
 ...

****************************************

             Resistance        Transition
Input Port   Rise     Fall     Rise     Fall
------------------------------------------------------------
A            --       --       --       --


                Driving Cell
Input Port   Rise             Fall             Mult    Attrs
------------------------------------------------------------------------------
A            AND              AND              --
```

SEE ALSO

all_inputs(2)
remove_driving_cell(2)
report_ports(2)
reset_design(2)
set_drive(2)
set_input_transition(2)
set_max_capacitance(2)
set_min_capacitance(2)

set_eco_placement_net_weight

Set different net weight for individual net as user-defined net weight priority.

SYNTAX

status **set_eco_placement_net_weight**
-nets collection_of_net -weight weight
| *-reset*

Data Types

collection_of_net list or collection
weight integer

ARGUMENTS

-nets *collection_of_net*

Set net weight for one or more nets and store them in memory. The two options *-nets* and *-weight* must be used together. The option *-nets* can not be used with *-reset*. They are mutually exclusive.

-weight *weight*

Specifies the net weight. The value is integer type and should be great than or equal to 1. If the option is not used, the default net weight is 1. The two options -nets and -weight must be used together.

The net weight represents the spring force between eco cells. The bigger the net weight, the higher the force. Currently the command **place_eco_cells** uses connectivity based coarse placement. By default, all net weight are same. So the eco cell is placed with even distance to driver and load. User can decide how close the eco cells by changing the net weight.

-reset

Clean up the internal net weight mapping table. All net weight is reset to 1. The option *-nets* can not be used with *-reset*. They are mutually exclusive.

DESCRIPTION

This command sets different net weight for individual net as user-defined net weight priority. It is used before *place_eco_cells -honor_user_net_weight*, that performs placement based on user-defined net weight setting. Please refer to the man page of place_eco_cells for the flow usage about user-defined net weight based coarse placement.

The second *set_eco_placement_net_weight -nets -weight* command overwrites the first one. If user adjusts the net weight before but wants to roll back these nets, he can do it like below. *set_eco_placement_net_weight -nets $n -weight 1*

Close_block and close_lib will internally clean up the net weight data in memory since the net weight setting will confuse the placement in next session and the data may be dirty for the next opened design.

If user wants to reuse the setting, he can use report_eco_placement_net_weight -output to dump the setting before close_block and close_lib and load it when necessary.

The three commands set_eco_placement_net_weight, report_eco_placement_net_weight and place_eco_cells -honor_user_net_weight will also check if the current lib and design is consistent with the lib and design stored in memory originally, if not, the command will clean up the net weight data in memory first and then apply the command to avoid inconsistent data and unexpected result.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets net weight.

```
prompt> set_eco_placement_net_weight \
    -nets $netList -weight 3
```

The following example resets net weight.

```
prompt> set_eco_placement_net_weight -reset
```

SEE ALSO

place_eco_cells(2)
report_eco_placement_net_weight(2)

set_eco_power_intention

This command updates power intention for ECO buffers.

SYNTAX

status **set_eco_power_intention**
[-cells *cells*]

Data Types

cells collection or list

ARGUMENTS

-cells *cells*

Specifies ECO cells to update power intention. The specified cells and cells in the **eco buffer trees** of the specified cells are target cells of this command. Here, **eco buffer trees** means a group of connected timing ECO cells, separated by cells of other types. If this option is not specified, all cells added by timing ECO commands will be updated.

DESCRIPTION

This command updates power intention for timing ECO buffers (or inverters) in primary voltage area. Here **timing eco buffers (or inverters)** means buffers (or inverters) added by create_cell and timing ECO commands, including **add_buffer**, **add_eco_repeater**, **split_fanout**, **add_buffer_on_route** and **size_cell** (in freeze_silicon).

The power intention includes supply net information and power domain information. This command will update supply nets for target cells. And if the power domain inherited from the hierarchy of a target cell is not consistent with the voltage area of this cell, the command will resolve the inconsistency issue by applying a physical feedthrough (PFT) or logical feedthrough (LFT) solution.

The buffer added on a physical feedthrough net will have the inconsistency issue that logically it belongs to the logic hierarchy of the net while physically it belongs to the voltage area where it is placed. Here the physical feedthrough net is a net which passes through a foreign voltage area in the physical view of its design.

The physical feedthrough (PFT) solution resolves this inconsistency by setting one power domain of the voltage area directly on the buffer (power domain on instance). And the logical feedthrough (LFT) solution resolves this inconsistency by moving the buffer to a logic hierarchy under a power domain of the voltage area. This command will try PFT solution first, and if PFT can not work, the LFT solution will be used if enabled by the settings.

The PFT solution is always enabled in this command. The LFT solution is not enabled by default and can be enabled or not by the command **create_voltage_area_rule** and the app option **mv.cells.mv_allow_logical_feedthrough**.

- When there is voltage area rule with the option **-allow_logical_feedthrough true**, the LFT solution is allowed without considering the app option **mv.cells.mv_allow_logical_feedthrough**.
- When there is voltage area rule with the option **-allow_logical_feedthrough false**, The LFT solution is not allowed without considering the app option **mv.cells.mv_allow_logical_feedthrough**.
- When there is no voltage area rule, the LFT solution is allowed if the app option **mv.cells.mv_allow_logical_feedthrough** is **true** and not allowed if the app option **mv.cells.mv_allow_logical_feedthrough** is **false**.

If there is any violation when updating power intention for a target cell, the power intention of the whole buffer tree that contains this cell will not be updated. This command can handle multiple buffer trees. If the processing for some trees fails during update power intention, the cells of the failed trees will be added to the collection $ecoUpdateSupplyFailedCells and a warning message is issued for each failing buffer tree. But these cells will not be removed by this command.

This command does not support undo.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples shows how to using the **set_eco_power_intention** command to update supply nets and power domain on instance for the specified cells.

```
prompt> set_eco_power_intention -cells {eco_cell_0}
```

The following examples shows how to enable LFT solution in the **set_eco_power_intention** command.

```
prompt> create_voltage_area_rule -name vaRule -allow_logical_feedthrough true
prompt> set_eco_power_intention -cells {eco_cell_0}
```

SEE ALSO

add_buffer_on_route(2)
check_mv_design(2)
create_voltage_area_rule(2)

set_edit_setting

Sets common editing settings.

SYNTAX

string **set_edit_setting** [-ignore_fixed *state*]
[-ignore_locked *state*]
[-keep_pin_on_edge *state*]
[-self_intersection *state*]
[-update_floorplan *state*]
[-update_color_mask *state*]
[-auto_display_hidden *state*]
[-select_partial_object *state*]
[-select_edge *state*]
[-select_vertex *state*]
[-select_center_line *state*]
[-select_center_vertex *state*]
[-expand_hit_macro_cell *state*]
[-expand_hit_blockage *state*]
[-expand_hit_constraint *state*]
[-honor_ndr *state*]
[-hierarchical_routing *state*]
[-pin_layer_policy *layer_policy*]
[-specified_pin_layer *layer_name*]
[-default]

Data Types

state string or collection
layer_policy string or collection
layer_name string or collection

ARGUMENTS

-ignore_fixed *state*

Ignores the locks flag and modify fixed or locked objects. The option is synonym of **-ignore_locked** option.

-ignore_locked *state*

Ignores the locks flag and modify fixed or locked objects. The option is synonym of **-ignore_fixed** option.

-keep_pin_on_edge *state*

Keeps pins on parent edge of the pin.

-self_intersection *state*

Allows self intersection of shapes.

-update_floorplan *state*

Updates the floorplan when the cell boundary is changed. Pin locations are updated based on the change in cell boundary. If the changes are small or do not affect the original placement of the pins, the pins are moved only a small distance or not at all. When the change to cell boundary is large, the pins locations are updated to be useful in displaying connections, but you are expected to place the pins again pins at a later time.

-update_color_mask *state*

Updates the color mask for net shapes to match the tracks they are snapped to, if the track has a specified mask color.

-auto_display_hidden

Displays hidden objects automatically.

-select_partial_object *state*

Allows selection of an object by selecting only a part of the object, such as a vertex or edge.

-select_edge *state*

Allows objects to be selected by selecting only the edge of the object.

-select_vertex *state*

Allows objects to be selected by selecting only a vertex of the object.

-select_center_line *state*

Allows objects to be selected by selecting only the center line of the object.

-select_center_vertex *state*

Allows objects to be selected by selecting only the center vertex of the object.

-expand_hit_macro_cell *state*

Expands the hit type to include macro cells.

-expand_hit_blockage *state*

Expands the hit type to include blockages.

-expand_hit_constraint *state*

Expands the hit type to include constraints such as bounds, voltage areas, and so on.

-honor_ndr *state*

Honor non-default width and spacing during editing.

-hierarchical_routing *state*

Treats hierarchical nets as the same net for routing and tracing.

-pin_layer_policy *layer_policy*

Specifies the layer policy applied while moving pins between edges. Valid values are: **preferred**, **keep**, and **specified**.

- **preferred**: Determine the pin layer based on the preferred routing direction going through the edge. This is the default

setting.

- **keep**: Use the same layer
- **specified**: Use the specified layer

-specified_pin_layer *layer_name*

Specifies the pin layer name when the layer policy setting is "specified"

-default

Resets all options to their defaults.

DESCRIPTION

This command is used to specify common editing settings.

EXAMPLES

The following example enables self intersection of shapes while performing edits.

```
prompt> set_edit_setting -self_intersection true
1
```

SEE ALSO

get_edit_setting(2)

set_editability

Changes the hierarchical edit control of specified blocks.

SYNTAX

int **set_editability**
[-blocks *list_of_blocks*]
[-from_level *depth*]
[-to_level *depth*]
[-value *true* | *false*]
[-verbose]

Data Types

list_of_blocks string or collection
depth integer

ARGUMENTS

-blocks *list_of_blocks*

The hierarchical edit control of specified blocks would be changed with respect to the current top-level context. If this option is not specified, the default behavior shall change the hierarchical editability of the current_block.

-from_level *depth*

Specifies the design depth from which editability value is being specified. Thus it is applicable for hierarchies at the specified design depth and below. This option is mutually exclusive with the *-to_level* and *-blocks* options.

-to_level *depth*

Specifies the design depth to which editability value is being specified. Thus it is applicable for hierarchies at the specified design depth and above. This option is mutually exclusive with the *-from_level* and *-blocks* options.

-value *true* | *false*

A true value makes the specified blocks editable and false value makes the specified blocks uneditable. By default, this option is true.

-verbose

Display verbose information.

DESCRIPTION

This command changes the hierarchical edit control of specified blocks and returns an integer indicating the number of blocks with changed hierarchical edit controls.

EXAMPLES

The following example sets the hierarchical edit control of a single block to mark it as editable:

```
prompt> set_editability -blocks blk1
1
```

The following example sets the hierarchical edit control of multiple blocks to mark them as uneditable (read-only):

```
prompt> set_editability -blocks [get_blocks {blk1 blk2 blk3}] -value false
3
```

Let's assume a MPH design with 3 levels: top, mid and bot. The following example sets the hierarchical edit control for all levels of physical hierarchies to mark them as uneditable (read-only):

```
prompt> set_editability -from_level 0 -value false -verbose
Block lib:top.design is set uneditable (read-only) in the top-level context of top.
Block lib:mid.design is set uneditable (read-only) in the top-level context of top.
Block lib:bot.design is set uneditable (read-only) in the top-level context of top.
3
```

The following example sets the hierarchical edit control for all levels of physical hierarchies except top to mark them as editable:

```
prompt> set_editability -from_level 0 -value true
prompt> set_editability -to_level 0 -value false
```

SEE ALSO

get_editability(2)
report_editability(2)

set_edrc_setting

Sets common Editor DRC settings.

SYNTAX

status **set_edrc_setting**
[-name *string*]
[-value true | false]
[-rules *name_value_list*]
[-enclosed_via_spacing_rule true | false]
[-general_via_spacing_rule true | false]
[-minimum_edge_rule true | false]
[-minimum_length_and_area_rule true | false]
[-rdl_acute_angle_rule true | false]
[-rdl_right_angle_rule true | false]
[-via_density_rule true | false]
[-honor_ndr true | false]
[-dpt_odd_cycle true | false]
[-dpt_precolor true | false]
[-end_of_line_spacing_rule true | false]
[-via_enclosure_rule true | false]
[-metal_width_rule true | false]
[-metal_span_spacing_rule true | false]
[-filter_same_net_spacing true | false]
[-check_fill true | false]
[-max_error_limit *number*]
[-max_shape_limit *number*]
[-max_processing_time *number*]
[-show_error_browser true | false]
[-check_drc true | false]
[-default]

Data Types

number int

ARGUMENTS

-name *string*

Specifies the name of the rule to set with the -value argument

-value true | **false**

Controls whether the Editor DRC checks the rule specified by the -name argument

-rules *name_value_list*

Specifies a list of rule name value pairs to set. Mutually exclusive with the -name argument.

-enclosed_via_spacing_rule true | false

Controls whether Editor DRC checks the enclosed via spacing rule.

-general_via_spacing_rule true | false

Controls whether Editor DRC checks the general via spacing rule.

-minimum_edge_rule true | false

Controls whether Editor DRC checks the following minimum edge rules: maximum total number of short edge, maximum total short edge length, convex-concave minimum edge length, adjacent minimum edge length, and special notch.

-minimum_length_and_area_rule true | false

Controls whether Editor DRC checks the minimum length and general minimum area rules.

-rdl_acute_angle_rule true | false

Controls whether Editor DRC checks for acute angle violations for RDL routes.

-rdl_right_angle_rule true | false

Controls whether Editor DRC checks for right angle violations for RDL routes.

-via_density_rule true | false

Controls whether Editor DRC checks the via density rule.

-honor_ndr true | false

Controls whether Editor DRC checks the following nondefault routing rules: metal spacing and via spacing.

-dpt_odd_cycle true | false

Controls whether Editor DRC checks for the existence of double-patterning odd cycles.

-dpt_precolor true | false

Controls whether Editor DRC checks for double-patterning spacing violations for colored shapes.

-end_of_line_spacing_rule true | false

Controls whether Editor DRC checks the following end-of-line spacing rules: end-of-line to end-of-line spacing rule, one-neighbor end-to-end table-based spacing rule, one-neighbor end-of-line spacing rule, and two-neighbor end-of-line spacing rule.

-via_enclosure_rule true | false

Controls whether Editor DRC checks the following via enclosure rules: minimum enclosure rule, zero-neighbor end-of-line enclosure rule, and T-shape metal extension end-of-line enclosure rule.

-metal_width_rule true | false

Controls whether Editor DRC checks the following metal width rules: discrete metal width rule and signal routing maximum metal width rule.

-metal_span_spacing_rule true | false

Controls whether Editor DRC checks the following metal span spacing rules: metal span spacing and metal span orthogonal spacing rule.

-filter_same_net_spacing true | false

Controls whether Editor DRC checks for spacing violations between shapes of the same net.

-check_fill true | false

Controls whether Editor DRC checks shapes inside fill cells.

-max_error_limit *number*

Specifies the maximum number of DRC errors to report. The default is 200.

-max_shape_limit *number*

Specifies the maximum number of shapes for DRC checking. The default is 400.

-max_processing_time *number*

Specifies the maximum time (in seconds) for DRC checking. The default is 10.

-show_error_browser true | false

Controls whether the error browser is displayed when errors are detected.

-check_drc true | false

Controls whether Editor DRC is enabled during interactive editing.

-default

Resets all options to their defaults.

DESCRIPTION

This command specifies common Editor DRC settings.

By default, the editor checks shorts and the following spacing rules: minimum spacing, via corner spacing, u-shape spacing, fat metal spacing and minimum width, net-type based spacing, and via farm spacing.

You can use this command to enable and disable additional supported rules and other controls for Editor DRC.

EXAMPLES

The following example enables odd cycle design rule checking while performing edits.

```
prompt> set_edrc_setting -name {DPT Odd Cycle} -value {true}
1
```

The following example enables short and minimum spacing checking while performing edits.

```
prompt> set_edrc_setting -rules {{Short} {true} {Minimum Spacing} {false}}
1
```

SEE ALSO

get_edrc_setting(2)

set_equivalent

Declare that supply nets or supply sets are electrically or functionally equivalent.

SYNTAX

string **set_equivalent**
  [-function_only]
  [-nets *supply_net_list*]
  [-sets *supply_set_list*]

Data Types

supply_net_list list
supply_set_list list

ARGUMENTS

-function_only

Specifies that the supplies are functionally equivalent rather than electrically equivalent.

-nets *supply_net_list*

A list of supply port and/or supply net names that are equivalent.

-sets *supply_set_list*

A list of supply set names that are equivalent.

DESCRIPTION

The **set_equivalent** command declares that two or more supplies are equivalent. If **-function_only** is specified, then the supplies are declared to be functionally equivalent only; otherwise the supplies are declared to be electrically equivalent, which implies that they are also functionally equivalent.

If **-nets** is specified, the command defines equivalence for a list of supply ports and/or supply nets. If **-sets** is specified, the command defines equivalence for a list of supply sets and/or supply set handles. One or the other of these options, but not both, shall be specified.

EXAMPLES

The following example declares the supply nets 'vdda' and 'vddb' to be electrically equivalent.

```
prompt> set_equivalent -nets {vdda vddb}
```

The following example declares the supply sets 'ss1' and 'ss2 to be electrically equivalent.

```
prompt> set_equivalent -sets {ss1 ss2} -function_only
```

SEE ALSO

create_supply_net(2)
create_supply_set(2)

set_extraction_options

Sets extraction options that influence extraction.

SYNTAX

string **set_extraction_options**
[-corners *corners*]
[-late_cap_scale *late_cap_scale*]
[-early_cap_scale *early_cap_scale*]
[-late_res_scale late_res_scale]
[-early_res_scale *early_res_scale*]
[-late_ccap_scale *late_ccap_scale*]
[-early_ccap_scale *early_ccap_scale*]
[-late_vr_horizontal_cap_scale *late_vr_horizontal_cap_scale*]
[-early_vr_horizontal_cap_scale *early_vr_horizontal_cap_scale*]
[-late_vr_vertical_cap_scale *late_vr_vertical_cap_scale*]
[-early_vr_vertical_cap_scale *early_vr_vertical_cap_scale*]
[-late_vr_horizontal_res_scale *late_vr_horizontal_res_scale*]
[-early_vr_horizontal_res_scale *early_vr_horizontal_res_scale*]
[-late_vr_vertical_res_scale *late_vr_vertical_res_scale*]
[-early_vr_vertical_res_scale *early_vr_vertical_res_scale*]
[-late_vr_via_res_scale *late_vr_via_res_scale*]
[-early_vr_via_res_scale *early_vr_via_res_scale*]
[-late_rde_cap_scale *late_rde_cap_scale*]
[-early_rde_cap_scale *early_rde_cap_scale*]
[-late_rde_res_scale *late_rde_res_scale*]
[-early_rde_res_scale *early_rde_res_scale*]
[-late_ccap_threshold *late_ccap_threshold*]
[-early_ccap_threshold *early_ccap_threshold*]
[-late_ccap_ratio *late_ccap_ratio*]
[-early_ccap_ratio *early_ccap_ratio*]
[-process_scale *process_scale*]
[-reference_direction vertical | horizontal | use_from_tluplus]
[-real_metalfill_extraction none | floating | grounded | auto]
[-virtual_metalfill_extraction none | floating | grounded]
[-virtual_metalfill_parameter_file *virtual_metalfill_parameter_file*]
[-virtual_metalfill_starrc_parameter_file *virtual_metalfill_starrc_parameter_file*]
[-virtual_shield_extraction true | false]
[-enable_ccap_or_filtering true | false]
[-include_pin_resistance true | false]
[-honor_mask_constraints true | false]
[-default]

Data Types

corners list
late_cap_scale float

early_cap_scale float
early_res_scale float
late_ccap_scale float
early_ccap_scale float
late_vr_horizontal_cap_scale float
early_vr_horizontal_cap_scale float
late_vr_vertical_cap_scale float
early_vr_vertical_cap_scale float
late_vr_horizontal_res_scale float
early_vr_horizontal_res_scale float
late_vr_vertical_res_scale float
early_vr_vertical_res_scale float
late_vr_via_res_scale float
early_vr_via_res_scale float
late_rde_cap_scale float
early_rde_cap_scale float
late_rde_res_scale float
early_rde_res_scale float
late_ccap_threshold float
early_ccap_threshold float
late_ccap_ratio float
early_ccap_ratio float
process_scale float
virtual_metalfill_parameter_file string
virtual_metalfill_starrc_parameter_file string

ARGUMENTS

-corners *corners*

Specifies the list of corners for which to apply the extraction options.

-late_cap_scale *late_cap_scale*

Specifies the late capacitance scaling factor. Will be deprecated and scales will be supported by set_paraisitcs_derate.

-early_cap_scale *early_cap_scale*

Specifies the Early capacitance scaling factor. Will be deprecated and scales will be supported by set_paraisitcs_derate.

-late_res_scale late_res_scale

Specifies the late resistance scaling factor. Will be deprecated and scales will be supported by set_paraisitcs_derate.

-early_res_scale *early_res_scale*

Specifies the early resistance scaling factor. Will be deprecated and scales will be supported by set_paraisitcs_derate.

-late_ccap_scale *late_ccap_scale*

Specifies the late coupling capacitance scaling factor. Will be deprecated and scales will be supported by set_paraisitcs_derate.

-early_ccap_scale *early_ccap_scale*

Specifies the early coupling capacitance scaling factor. Will be deprecated and scales will be supported by

set_paraisitcs_derate.

-late_vr_horizontal_cap_scale ***late_vr_horizontal_cap_scale***

Specifies the late virtual route (vr) horizontal capacitance scaling factor.

-early_vr_horizontal_cap_scale ***early_vr_horizontal_cap_scale***

Specifies the early virtual route (vr) horizontal capacitance scaling factor.

-late_vr_vertical_cap_scale ***late_vr_vertical_cap_scale***

Specifies the late virtual route (vr) vertical capacitance scaling factor.

-early_vr_vertical_cap_scale ***early_vr_vertical_cap_scale***

Specifies the early virtual route (vr) vertical capacitance scaling factor.

-late_vr_horizontal_res_scale ***late_vr_horizontal_res_scale***

Specifies the late virtual route (vr) horizontal resistance scaling factor.

-early_vr_horizontal_res_scale ***early_vr_horizontal_res_scale***

Specifies the early virtual route (vr) horizontal resistance scaling factor.

-late_vr_vertical_res_scale ***late_vr_vertical_res_scale***

Specifies the late virtual route (vr) vertical resistance scaling factor.

-early_vr_vertical_res_scale ***early_vr_vertical_res_scale***

Specifies the early virtual route (vr) vertical resistance scaling factor.

-late_vr_via_res_scale ***late_vr_via_res_scale***

Specifies the late virtual route (vr) via resistance scaling factor.

-early_vr_via_res_scale ***early_vr_via_res_scale***

Specifies the early virtual route (vr) via resistance scaling factor.

-late_rde_cap_scale ***late_rde_cap_scale***

Specifies the late RDE capacitance scaling factor which is applied on top of late_cap_scale. Will be deprecated and scales will be supported by set_paraisitcs_derate.

-early_rde_cap_scale ***early_rde_cap_scale***

Specifies the early RDE capacitance scaling factor which is applied on top of early_cap_scale. Will be deprecated and scales will be supported by set_paraisitcs_derate.

-late_rde_res_scale late_rde_res_scale

Specifies the late RDE resistance scaling factor which is applied on top of late_res_scale. Will be deprecated and scales will be supported by set_paraisitcs_derate.

-early_rde_res_scale ***early_rde_res_scale***

Specifies the early RDE resistance scaling factor which is applied on top of early_res_scale. Will be deprecated and scales will be supported by set_paraisitcs_derate.

-late_ccap_threshold ***late_ccap_threshold***

Specifies the late coupling capacitance filtering threshold. The units are user units. Default is 0.003 pF.

-early_ccap_threshold *early_ccap_threshold*

Specifies the early coupling capacitance filtering threshold. The units are user units. Default is 0.003 pF.

-late_ccap_ratio *late_ccap_ratio*

Specifies the late coupling capacitance filtering ratio. The default is 0.03 times the total capacitance.

Only *late_ccap_threshold* and *late_ccap_ratio* are used as global settings for filtering. The coupling capacitance between two nets is filtered, when both *late_ccap_threshold* and *late_ccap_ratio* conditions are satisfied.

For example, consider two coupled nets: netA and netB. The lumped coupling capacitance between them is 1ff, total capacitance (including pin capacitance) of netA is 50ff, and the total capacitance of netB is 20ff. The default ccap_threshold and ccap_ratio are used. Here is the filtering calculation, where "CC" represents coupling capacitance:

```
Example:
  CC(netA to netB) = 1
  TotalCap(netA)   = 50
  TotalCap(netB)   = 20

 Comparing ccap_threshold and cc_ratio:
  CC(netA to netB)          = 1 < 3            --> TRUE
  CC(netA to netB) / TC(netA) = 1/50 = 0.02 < 0.03 --> TRUE
  CC(netA to netB) / TC(netB) = 1/20 = 0.05 < 0.03 --> FALSE
```

Based on this calculation, this coupling pair is not filtered because the ratio of coupling capacitance (1) to total capacitance for netB (20) is 0.05, which is greater than the default of 0.03.

-early_ccap_ratio *early_ccap_ratio*

Specifies the early coupling capacitance filtering ratio. The default is 0.03 of the total capacitance.

-process_scale *process_scale*

Specifies the process scaling factor. The default is 1.0. Value should be specified greater than 0 and less than or equal to 1.0. If there is HALF_NODE_SCALE_FACTOR specified in ITF then extractor will honor ITF's HALF_NODE_SCALE_FACTOR.

-reference_direction vertical | horizontal | use_from_tluplus

Specifies the reference direction to which to apply the orientation-based etch tables. The reference direction applies to all corners. The default is **use_from_tluplus**.

- **vertical** - Specifies that the reference direction is vertical when applying the orientation-based etch tables, regardless of the **REFERENCE_DIRECTION** setting in the TLUPlus files.
- **horizontal** - Specifies that the reference direction is horizontal when applying the orientation-based etch tables, regardless of the **REFERENCE_DIRECTION** setting in the TLUPlus files.
- **use_from_tluplus** - Specifies that the reference direction is defined by the **REFERENCE_DIRECTION** setting in the TLUPlus files. The TLUPlus files are specified by the **set_parasitic_parameters** command.

-real_metalfill_extraction none | floating | grounded | auto

Specifies the models that can be used to extract metal fill. Metal fill are metal shapes that are inserted during chip finishing to create a more uniform metal density across the chip. When metal fill is inserted, the capacitance effect can be extracted based on the filler type: floating or grounded. To ignore the capacitance effects of metal fill, specify "none". By default, the setting is "none". When setting is "auto", metal fill type will be automatically detected and effects applied accordingly.

-virtual_metalfill_extraction none | floating | grounded

Specifies the models that can be used to virtual metal fill extraction. In ICC2, pre-route and post-route extractor emulate virtual metalfill based on the given virtual metal file parameter files for RC extraction. When metal fill is emulated, the capacitance effect can be extracted based on the filler type: floating or grounded. To ignore the capacitance effects of metal fill, specify "none". By default, the setting is "none".

-virtual_metalfill_parameter_file fileName

Specifies metal fill parameters used for ICC2 virtual metalfill emulation in RC extraction. The syntax of parameter file is: <db_layer_name A> <fill_widthA> <fill_spaceA> <db_layer_name B> <fill_widthB> <fill_spaceB> ..... <db_layer_name N> <fill_widthN> <fill_spaceN>

-virtual_metalfill_starrc_parameter_file fileName

Specifies metal file paramters used for StarRC inDesign virtual metalfill RC extracton. It is optional to use this file for ICC2 extractor virtual metalfill RC extraction, if "-virtual_metalfill_parameter_file fileName" is not specified. But only use fill_width, fill_route_w_spacing parameters in ICC2 extractor. The syntax of parameter file is: <db_layer_name A> <directionA> <fill_widthA> <min_fill_lengthA> <max_fill_lengthA> <fill_route_w_spacingA> <fill_route_l_spacingA> <fill_fill_w_spacingA> <fill_fill_l_spacingA> <fill_blockage_w_spacingA> <fill_blockage_l_spacingA> <fill_ndr_w_spacingA> <fill_ ndr _l_spacingA> <fill_chip_w_spacingA> < fill_chip_l_spacingA> <fill_blockage_excluded_cellsA>

<db_layer_name B> <directionB> <fill_widthB> <min_fill_lengthB> <max_fill_lengthB> <fill_route_w_spacingB> <fill_route_l_spacingB> <fill_fill_w_spacingB> <fill_fill_l_spacingB> <fill_blockage_w_spacingB> <fill_blockage_l_spacingB> <fill_ndr_w_spacingB> <fill_ ndr _l_spacingB> <fill_chip_w_spacingB> < fill_chip_l_spacingB> < fill_blockage_excluded_cellsB> ..... <db_layer_name N> <directionN> <fill_widthN> <min_fill_lengthN> <max_fill_lengthN> <fill_route_w_spacingN> <fill_route_l_spacingN> <fill_fill_w_spacingN> <fill_fill_l_spacingN>> <fill_blockage_w_spacingN> <fill_blockage_l_spacingN> <fill_ndr_w_spacingN> <fill_ ndr _l_spacingN> <fill_chip_w_spacingN> < fill_chip_l_spacingN> <fill_blockage_excluded_cellsN>

-virtual_shield_extraction true | false

This option applies only to postroute extraction. When this option is set to **true** (the default), the tool performs virtual shield extraction by considering the shielding rules. Otherwise, the shielding rules are not considered during extraction. Shielding rules are nondefault routing rules that are defined by using the **create_routing_rule** command. Both the nets with nondefault routing rules and their neighbors might see their RC values affected due to shielding effects when this option is set to **true**.

-enable_ccap_or_filtering true | false

Enables OR-style coupling capacitance filtering. Using this filtering, the coupling capacitance is filtered when one or both of the *ccap_threshold* and *ccap_ratio* settings are satisfied.

The default style is AND, which means that coupling capacitance is filtered when both the *ccap_threshold* and *ccap_ratio* settings are satisfied.

-include_pin_resistance true | false

Creates a resistor network for a pin with multiple connections, when set to **true**. There is only a resistance value within the pin, and no capacitance. This option can be applied when performing extraction on detail-routed designs. The default is **true**.

-honor_mask_constraints true | false

The default is **false**, which causes mask constraint (color) attributes to be ignored. When this option is set to **true**, the tool considers mask constraints during extraction. Colors of both victim and aggressor nets are considered. The design should have geometries with mask (color) attributes annotated. The TLUPlus file should define mask-dependent (color-dependent) etch-versus-width-spacing (EVWS) information.

-operating_frequency *operating_frequency*

Specifies the operating frequency value used for 3DIC. The unit is Hz.

-default

Resets all options to their defaults.

DESCRIPTION

This command sets extraction options used during RC extraction. The scaling options will be associated with default constraint corner, or ea list of constraint corners. Other options are global and not corner specific.

The default scaling factor value is 1.0, which means no scaling. This command supports 0 and positive scaling factor value, and checks and warns when abnormal negative scaling factors are specified.

The default filtering threshold is 3ff and 3 percent. If the total coupling capacitance between a net pair is less than 3ff and 3 percent of total capacitance of both nets, all coupling entries between the net pair are filtered.

Multicorner-Multimode Support

EXAMPLES

In the following example, extraction options are applied to the current constraint corner.

```
prompt> set_extraction_options -late_cap_scale 1.1 -early_cap_scale 1.05
1
```

SEE ALSO

all_corners(2)
create_corner(2)
create_mode(2)
current_corner(2)
get_corners(2)
remove_corners(2)
report_extraction_options(2)
report_parasitic_parameters(2)
set_parasitic_parameters(2)
set_parasitics_derate(2)
reset_parasitics_derate(2)
report_parasitics_derate(2)
get_user_units(2)

set_false_path

Identifies paths in a design to mark as false, so that they are not considered during timing analysis.

SYNTAX

status **set_false_path**
[-setup] [-hold]
[-rise] [-fall]
[-reset_path]
[-from *from_list*
| -rise_from *rise_from_list*
| -fall_from *fall_from_list*]
[-through *through_list*]
[-rise_through *rise_through_list*]
[-fall_through *fall_through_list*]
[-to *to_list*
| -rise_to *rise_to_list*
| -fall_to *fall_to_list*]
[-comment *comment*]

Data Types

from_list list
rise_from_list list
fall_from_list list
through_list list
rise_through_list list
fall_through_list list
to_list list
rise_to_list list
fall_to_list list
comment list

ARGUMENTS

-setup

Marks setup (maximum) paths as false. This option disables setup checking for specified paths. If you do not specify either **-setup** or **-hold**, both setup and hold timing are marked false.

-hold

Marks hold (minimum) paths as false. This option disables hold checking for specified paths. If you do not specify either **-setup** or **-hold**, both setup and hold timing are marked false.

-rise

Marks rising delays as false, as measured on the path endpoint. If you do not specify either **-rise** or **-fall**, both rise and fall timing are marked false.

-fall

Marks falling delays as false, as measured on the path endpoint. If you do not specify either **-rise** or **-fall**, rise and fall timing are marked false.

-reset_path

Removes existing point-to-point exception information from the specified paths. If used with **-to** only, all paths leading to the specified endpoints are reset. If used with **-from** only, all paths leading from the specified startpoints are reset. If used with **-from** and **-to**, only paths between those points are reset. Only information of the same rise/fall setup/hold type is reset. This is equivalent to using the **reset_path** command with similar arguments before the **set_false_path** command is issued.

-from ***from_list***

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If you specify a cell, one path startpoint on that cell is affected. You can use only one of the **-from**, **-rise_from**, or **-fall_from** options.

-rise_from ***rise_from_list***

Same as the **-from** option, except that the path must rise from the specified objects. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-from**, **-rise_from**, or **-fall_from** options.

-fall_from ***fall_from_list***

Same as the **-from** option, except that the path must fall from the specified objects. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-from**, **-rise_from**, or **-fall_from** options.

-through ***through_list***

Specifies a list of pins, ports, cells or nets through which the disabled paths must pass. You can specify **-through** more than one time in one command invocation. Nets are interpreted to imply the leaf-level driver pins.

If you do not specify any type of **through** option, all timing paths specified using the **-from** and **-to** options are affected.

For a detailed discussion of the use of multiple **-through**, **-rise_through**, and **-fall_through** options, see the DESCRIPTION section.

-rise_through ***rise_through_list***

Specifies a list of pins, ports, cells or nets through which the disabled paths must pass. The paths must have a rising transition at the through points.

If you do not specify any type of **-through** option, all timing paths specified using the **-from** and **-to** options are affected.

For a detailed discussion of the use of multiple **-through**, **-rise_through**, and **-fall_through** options, see the DESCRIPTION section.

-fall_through ***fall_through_list***

Specifies a list of pins, ports, cells or nets through which the disabled paths must pass. The paths must have a falling transition at the through points.

If you do not specify any type of **-through** option, all timing paths specified using the **-from** and **-to** options are affected.

For a detailed discussion of the use of multiple **-through**, **-rise_through**, and **-fall_through** options, see the DESCRIPTION section.

-to ***to_list***

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If you specify a cell, one path endpoint on that cell is affected.

You can use only one of the **-to**, **-rise_to**, or **-fall_to** options.

-rise_to ***rise_to_list***

Specifies a list of timing path endpoint objects. but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source. The command takes into account any logical inversions along the clock path. You can use only one of the **-to**, **-rise_to**, or **-fall_to** options.

-fall_to ***fall_to_list***

Specifies a list of timing path endpoint objects, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the **-to**, **-rise_to**, or **-fall_to** options.

-comment ***comment***

Attaches a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

DESCRIPTION

The **set_false_path** command marks startpoint and endpoint pairs as false timing paths, that is, paths that cannot propagate a signal. The command removes timing constraints on these false paths, so that they are not considered during timing analysis. Path startpoints are input ports or register clock pins; path endpoints are register data pins or output ports. The command disables maximum delay (setup) checking and minimum delay (hold) checking for the specified paths.

The **set_false_path** command is a point-to-point timing exception command, and overrides the default single-cycle timing relationship for one or more timing paths. False path information always takes precedence over multicycle path information. Also, **set_false_path** commands override **set_max_delay** and **set_min_delay** commands.

You can use multiple **-through**, **-rise_through**, and **-fall_through** options in a single command to specify paths that traverse multiple points in the design. The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

-from A1 -through B1 -through C1 -to D1

If more than one object is specified within one **-through**, **-rise_through**, or **-fall_through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

-from A1 -through {B1 B2} -through {C1 C2} -to D1

To disable the timing at a particular cell along a path, use the **set_disable_timing** command.

To remove the false path designations set by **set_false_path**, use the **reset_paths** command.

For crosstalk analysis the false paths are treated as sensitizable. So, the aggressors on the false paths could still effect victim nets.

When the clocks are asynchronous to each other, use the **set_clock_groups -asynchronous** command instead of **set_false_paths** between the asynchronous clocks. Without the **set_clock_groups -asynchronous** command, the clocks will be treated as synchronized with each other.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example disables timing paths from ff12 to ff34.

```
prompt> set_false_path -from ff12 -to ff34
```

The following example disables rising timing paths from U14/Z to ff29/Reset.

```
prompt> set_false_path -rise_from U14/Z -to ff29/Reset
```

The following examples disables hold checking for endpoints clocked by CLK1.

```
prompt> set_false_path -hold -to [get_clocks CLK1]
```

The following example disables all timing paths from ff1/CP to ff2/D that pass through one or more of {U1/Z U2/Z} and one or more of {U3/Z U4/C}.

```
prompt> set_false_path -from ff1/CP -through {U1/Z U2/Z} \
  -through {U3/Z U4/C} -to ff2/D
```

The following example disables all timing paths from ff1/CP to ff2/D that rise through one or more of {U1/Z U2/Z} and fall to one or more of {U3/Z U4/C}.

```
prompt> set_false_path -from ff1/CP \
  -rise_through {U1/Z U2/Z} \
  -fall_through {U3/Z U4/C} -to ff2/D
```

SEE ALSO

current_design(2)
reset_design(2)
reset_paths(2)
set_clock_groups(2)
set_disable_timing(2)
set_max_delay(2)
set_min_delay(2)
set_multicycle_path(2)

set_fanout_load

Sets fanout_load for output ports in the current design.

SYNTAX

string **set_fanout_load** *value port_list*

float *value*
list *port_list*

ARGUMENTS

value

Shows **fanout_load** value, in units consistent with the **fanout_load** and **max_fanout** values in the technology library.

port_list

Provides a list of output ports.

DESCRIPTION

Specifies a **fanout_load** for output ports in the current design. By default, ports are considered to have fanout_load of 0.0. Fanout load for a net is the sum of **fanout_load** attributes for all input pins and output ports connected to the net. Output pins may have maximum fanout limits, specified in the library.

report_port -design_rule shows port **fanout_load** values.

To remove **fanout_load** information from ports, use **remove_fanout_load**.

Multicorner-Multimode Support

EXAMPLES

This example sets the fanout_load on ports matching "OUT*" to 3.0.

prompt> **set_fanout_load 3.0 "OUT*"**

SEE ALSO

remove_fanout_load(2)
report_port(2)

set_fix_multiple_port_nets

Sets the **fix_multiple_port_nets** attribute to a specified value on the current module or a list of modules.

SYNTAX

int **set_fix_multiple_port_nets**
-default | -all
[-feedthroughs]
[-outputs]
[-constants]
[-buffer_constants]
[-no_rewire]
[-exclude_clock_network]
[*object_list (design_list | module_list | block_list)*]

ARGUMENTS

-default

Removes the **fix_multiple_port_nets** attribute so that **compile** will use its default priority, that is no multiple port nets fixing.

-all

Inserts the same as **-feedthroughs -outputs -constants**. Note that logic constants are duplicated, not buffered. To fix constant port nets by buffering them, use the **-buffer_constants** option. Rewiring will be attempted before buffering.

-feedthroughs

Inserts buffers to isolate input ports from output ports at all levels of the hierarchy. Rewiring will be attempted before buffering.

-outputs

Inserts buffers so that no cell driver pin drives more than one output port at any level of the hierarchy. Rewiring will be attempted before buffering.

-constants

Duplicates logic constants so that no constant drives more than one output port at any level of the hierarchy. Rewiring will be attempted before constant duplication.

-buffer_constants

Buffers logic constants instead of duplicating them. Rewiring will be attempted before buffering.

-no_rewire

Disables a rewiring attempt before buffering multiple port nets and constant port nets. **-no_rewire** only applies to

fix_multiple_port_nets optimization. Use **set_boudary_optimization** to restrain other optimizations from rewiring.

-exclude_clock_network

Excludes nets belonging to the clock network from multiple port net fixing.

object_list

Specifies a list of designs or modules or blocks. The default is the current design.

DESCRIPTION

Sets the **fix_multiple_port_nets** attribute on a list of designs. If you do not specify a design list, the current design is used.

This attribute controls whether **compile** inserts extra logic into the design to ensure that there are no feedthroughs, or that there are no two output ports connected to the same net at any level of hierarchy.

The **set_fix_multiple_port_nets** command first attempts to rewire port nets. If no rewiring opportunity is found, then buffers will be inserted. Use **-no_rewire** to disable rewiring for the entire design or use **set_boudary_optimization** to disable rewiring on a specific hierarchy.

Certain three-state nets cannot be buffered, because this changes the logic functionality of the design.

If **set_fix_multiple_port_nets** is specified more than once on a design, the most recent setting is used and the earlier values are removed. To undo **set_fix_multiple_port_nets**, use the **-default** option.

To view the current setting of the **fix_multiple_port_nets** attribute, use **report_attribute** or **get_attribute**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the **fix_multiple_port_nets** attribute so that feedthroughs are buffered and constants are duplicated.

prompt> **set_fix_multiple_port_nets -feedthroughs -constants**

The following example sets the **fix_multiple_port_nets** attribute so that multiple outputs and constants are buffered.

prompt> **set_fix_multiple_port_nets -outputs -buffer_constants [get_modules *]**

The following example resets the **fix_multiple_port_nets** attribute so that no fixing is done.

prompt> **set_fix_multiple_port_nets -default**

SEE ALSO

compile(2)

current_design(2)
get_attribute(2)

set_fixed_objects

Sets the specified objects to the fixed state. The command sets the value of **physical_status** attribute to either **fixed** or **unrestricted**;

SYNTAX

status **set_fixed_objects**
[*object_list*]
[-unfix]

object_list collection or selection

ARGUMENTS

object_list

Collection or selection set containing objects to set or unset fixed status. If this option is not specified, global selection set is used.

-unfix

Specifies if given objects should be unfixed

DESCRIPTION

Set the fixed state for the supplied objects.

EXAMPLES

The following example fixes all selected objects

prompt> **set_fixed_objects**
1

The alternative syntax uses collection to specify objects.

prompt> **set_fixed_objects [get_selection]**
1

SEE ALSO

set_locked_objects(2)
get_selection(2)
change_selection(2)

set_floorplan_area_rules

Defines an area floorplan rule.

SYNTAX

set_floorplan_area_rules
-object_types *type_list*
-lib_cells *lib_cells*
-name *rule_name*
[-layers *layer_list*]
[-forbidden_list *area_list*]
[-forbidden_ranges {{*low high*} {*low1 high1*} ... }]
[-max *area*]
[-min *area*]
[-valid_list *area_list*]
[-valid_ranges {{*low high*} {*low1 high1*} ... }]

Data Types

type_list list
lib_cells collection
rule_name string
layer_list collection
area_list list
low float
high float
low1 float
high1 float
area float

ARGUMENTS

-object_types *type_list*

Specifies the list of object types for the area floorplan rule. The area of these type of objects will be checked. Valid values for this option are *block_boundary*, *core_area*, *hard_macro*, *placement_blockage*, *routing_blockage*, *shape*, *soft_macro* and *std_cell_area*. This option is mutually exclusive with **-lib_cells** and you must specify one or the other.

-lib_cells *lib_cells*

Specifies the collection of library cells for the area floorplan rule. The area of these library cells will be checked. This option is mutually exclusive with **-object_types** and you must specify one or the other.

-name *rule_name*

Specifies the name of the area floorplan rule. This is a mandatory option.

-layers ***layer_list***

Specifies the routing layers to be considered for the routing_blockage or shape object type. This option must be used together with **-object_types routing_blockage** or **-object_types shape**. This is an optional option.

-forbidden_list ***area_list***

Specifies a list of areas that are not allowed for the object or library cell. This option is mutually exclusive with **-valid_list**. Values specified cannot be negative. This is an optional option.

-forbidden_ranges {{*low high*} {*low1 high1*} ... }

Specifies a list of area ranges that are not allowed for the object or library cell. The area must not lie within any of *low* and *high* in the specified list of ranges. This option is mutually exclusive with **-valid_ranges**. Values specified cannot be negative. This is an optional option.

-max ***area***

Specifies the maximum area for the object or library cell. The area cannot be greater than this value. The value specified cannot be negative. If **-min** is also specified then this value must be greater than the min value. This is an optional option.

-min ***area***

Specifies the minimum area of object or library cell. The area cannot be less than this value. The value specified cannot be negative. If **-max** is also specified then this value must be lesser than the max value. This is an optional option.

-valid_list ***area_list***

Specifies a list of allowed areas for the area of the object or library cell. This option is mutually exclusive with **-forbidden_list**. Values specified cannot be negative. This is an optional option.

-valid_ranges {{*low high*} {*low1 high1*} ... }

Specifies a list of allowed area ranges for the object or library cell. The area must lie within any of *low* and *high* in the specified list of ranges. This option is mutually exclusive with **-forbidden_ranges**. Values specified cannot be negative. This is an optional option.

DESCRIPTION

The **set_floorplan_area_rules** command defines a named area floorplan rule in the current design. The defined rule is persistent. If another floorplan rule by the same name exists then the command errors out.

There is a difference between the object type *core_area* and *std_cell_area*. The *core_area* object type means core boundary region without cutting out any blockages and is typically applicable for top level whereas *std_cell_area* object type means core boundary region after cutting out all blockages and is typically applicable for block level.

If an area rule is defined for a library cell and another area rule is defined for a hard macro then the area rule defined for the library cell will take precedence over the area rule defined for hard macro when checks are done for that library cell.

If the measured value falls inside valid range or is a member of the valid list then there is no violation given by **check_floorplan_rules** regardless of other constraints like min, max, etc. If this measured value is outside valid range or list then a violation is reported if other constraints are specified and they are not met or if no other constraints are specified.

EXAMPLES

The following example creates an area rule named a1 to check the area of block boundary and hard macro. The area can't be greater than 2400 and can't lie between 370 and 790.

```
prompt> set_floorplan_area_rules -name a1 \
  -object_types {block_boundary hard_macro} -max 2400 \
  -forbidden_ranges {{370 790}}
```

SEE ALSO

remove_floorplan_rules(2)
report_floorplan_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_extension_rules(2)
set_floorplan_exception_rules(2)
set_floorplan_forbidden_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_length_rules(2)
set_floorplan_spacing_rules(2)
set_floorplan_width_rules(2)

set_floorplan_enclosure_rules

Defines a floorplan enclosure rule.

SYNTAX

set_floorplan_enclosure_rules
-from_object_types *from_type_list*
-to_object_types *to_type_list*
-to_lib_cells *lib_cells*
-sides *side_list*
-from_corner *corner_type*
-to_corner *corner_type*
-name *rule_name*
[-must_enclose]
[-follow_rotations]
[-ignore_rotate90]
[-layers *layer_list*]
[-forbidden_list *distance_list*]
[-forbidden_ranges {{*low high*} {*low1 high1*} ... }]
[-max *distance*]
[-min *distance*]
[-offset *distance*]
[-step *distance*]
[-valid_list *distance_list*]
[-valid_ranges {{*low high*} {*low1 high1*} ... }]

Data Types

from_type_list list
to_type_list list
lib_cells collection
side_list list
corner_type string
rule_name string
layer_list list
distance_list list
low float
high float
low1 float
high1 float
distance float

ARGUMENTS

-from_object_types *from_type_list*

Specifies the list of "from" object types for the enclosure floorplan rule. These type of objects will enclose the object types specified with **-to_object_types** or library cells specified with **-to_lib_cells**. Valid values for this option are *block_boundary*, *core_area*, *placement_blockage*, *routing_blockage*, *unplaceable_area* and *std_cell_area*. This is a mandatory option.

-to_object_types *to_type_list*

Specifies the list of "to" object types for the enclosure floorplan rule. These type of objects will be enclosed by other objects specified with **-from_object_types**. Valid values for this option are *core_area*, *hard_macro*, *placement_blockage*, *routing_blockage*, *soft_macro*, *unplaceable_area* and *std_cell_area*. This option is mutually exclusive with **-to_lib_cells** and you must specify one or the other.

-to_lib_cells *lib_cells*

Specifies the collection of library cells for the enclosure floorplan rule. These library cells are being enclosed by other objects specified in **-from_object_types**. This option is mutually exclusive with **-to_object_types** and either one of them must be specified.

-sides *side_list*

Specifies the sides or directions from which the "from" object encloses the "to" object. Spacing is checked between the objects. Valid values are *all*, *bottom*, *horizontal*, *left*, *right*, *top*, *vertical* and *manhattan*. The *horizontal* argument includes both *left* and *right*. Similarly, the *vertical* argument includes both *bottom* and *top*. This option is mutually exclusive with the **-from_corner** and **-to_corner** option pair. Either this option or **-from_corner** and **-to_corner** pair must be specified.

-from_corner *corner_type*

Specifies the corner from which the "from" object encloses the "to" object. Spacing is checked between the objects. Valid values are *all*, *bottom_left*, *bottom_right*, *top_left* and *top_right*. This option can be used when spacing must be checked from a corner instead of an edge. This option must be used together with **-to_corner** and is mutually exclusive with **-sides**.

-to_corner *corner_type*

Specifies the corner from which the "to" object is being enclosed by "from" object. Spacing is checked between the objects. Valid values are *all*, *bottom_left*, *bottom_right*, *top_left* and *top_right*. This option can be used when spacing must be checked from a corner instead of an edge. This option must be used together with **-from_corner** and is mutually exclusive with **-sides**.

-name *rule_name*

Specifies the name of the enclosure floorplan rule. This is a mandatory option.

-must_enclose

Specifies that the "from" object must completely enclose the "to" object from all sides. This is an optional option.

-follow_rotations

Specifies whether the sides specified by the **-sides** option should follow the rotations of library cells, that is, if the meaning of *horizontal* or *vertical* should change when the library cell has a 90-degree rotation. This option must be used together with **-to_lib_cells** or **-to_object_types hard_macro** or **-to_object_types soft_macro**. This is an optional option.

-ignore_rotate90

Specifies whether this rule can be ignored for library cells with 90-degree rotations. This option must be used together with **-to_lib_cells** or **-to_object_types hard_macro** or **-to_object_types soft_macro**. This is an optional option.

-layers *layer_list*

Specifies the routing layers to be considered for the *routing_blockage* object type. This option must be used together with **-from_object_types routing_blockage** or **-to_object_types routing_blockage**. This is an optional option.

-forbidden_list *distance_list*

Specifies a list of distances by which the "from" object cannot enclose the "to" object. This option is mutually exclusive with **-valid_list**. Values in the *distance_list* cannot be negative. This is an optional option.

-forbidden_ranges {{*low high*} {*low1 high1*} ... }

Specifies a list of distance ranges between which the "from" object cannot enclose the "to" object. The enclosing distance must not lie within any of *low* and *high* in the specified list of ranges. This option is mutually exclusive with **-valid_ranges**. Values cannot be negative. This is an optional option.

-max *distance*

Specifies the maximum distance by which the "from" object can enclose the "to" object. The distance cannot be greater than this value. The distance cannot be negative. If **-min** is also specified, this value must be greater than the min value. This is an optional option.

-min *distance*

Specifies the minimum distance by which the "from" object can enclose the "to" object. The distance cannot be less than this value. The value specified cannot be negative. If **-max** is also specified then this value must be less than the max value. This is an optional option.

-offset *distance*

Specifies a parameter in distance calculation between the "from" and "to" objects. This option must be used together with **-step**. This implies that the distance has to be an integer multiple of the **-step** value plus the **-offset** value. The value specified cannot be negative. This is an optional option.

-step *distance*

Specifies a parameter in distance calculation between the "from" and "to" objects. This option must be used together with **-offset**. This implies that the distance has to be an integer multiple of the **-step** value plus the **-offset** value. The value specified cannot be negative. This is an optional option.

-valid_list *distance_list*

Specifies a list of distances by which the "from" object can enclose the "to" object. This option is mutually exclusive with **-forbidden_list**. Values specified cannot be negative. This is an optional option.

-valid_ranges {{*low high*} {*low1 high1*} ... }

Specifies a list of distance ranges between which the "from" object can enclose the "to" object. The distance must lie within any of *low* and *high* in the specified list of ranges. This option is mutually exclusive with **-forbidden_ranges**. Values specified cannot be negative. This is an optional option.

DESCRIPTION

The **set_floorplan_enclosure_rules** command defines a named enclosure floorplan rule in the current design. The defined rule is persistent. If another floorplan rule by the same name exists then the command errors out.

There is a difference between the object types *core_area* and *std_cell_area*. The *core_area* object type is the core boundary region without cutting out any blockages and is typically applicable for the top level. The *std_cell_area* object type is the core boundary region after cutting out all blockages and is typically applicable for the block level.

If an enclosure rule is defined for a library cell and another enclosure rule is defined for a hard macro, the enclosure rule defined for the library cell takes precedence over the enclosure rule defined for hard macro when checks are done for that library cell.

If the measured value falls inside valid range or is a member of the valid list then there is no violation given by **check_floorplan_rules** regardless of other constraints like min, max, etc. If this measured value is outside valid range or list then a violation is reported if other constraints are specified and they are not met or if no other constraints are specified.

EXAMPLES

The following example creates an enclosure rule named e1. This rule checks the enclosing of hard macros and core area by the block boundary. The check will be done for horizontal sides, both left and right, and only at the bottom in vertical side. Must be greater than 2 and less than 100, but cannot be in the range between 5 and 15. Also, the distance must be an integer multiple of 2 plus 10.

```
prompt> set_floorplan_enclosure_rules -name e1 \
  -from_object_types block_boundary \
  -to_object_types {hard_macro core_area} -sides {bottom horizontal} \
  -forbidden_ranges {{5 15}} -valid_list {2 20 30} -max 100 -min 2 \
  -offset 10 -step 2
```

SEE ALSO

remove_floorplan_rules(2)
report_floorplan_rules(2)
set_floorplan_area_rules(2)
set_floorplan_extension_rules(2)
set_floorplan_exception_rules(2)
set_floorplan_forbidden_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_length_rules(2)
set_floorplan_spacing_rules(2)
set_floorplan_width_rules(2)

set_floorplan_exception_rules

Defines an exception floorplan rule in the design.

SYNTAX

set_floorplan_exception_rules
-rules *rule_list*
-name *rule_name*
[-from_object_types *type_list*]
[-to_object_types *type_list*]
[-object_types *type_list*]
[-from_lib_cells *lib_cells*]
[-to_lib_cells *lib_cells*]
[-lib_cells *lib_cells*]
[-identical]
[-orientation_types *orientation_list*]
[-reason *reason*]

Data Types

rule_list list
rule_name string
type_list list
lib_cells collection
orientation_list list
reason string

ARGUMENTS

-rules *rule_list*

Specifies the list of floorplan rules for which this exception rule needs to be applied. It is not required that rules specified in this list has to pre-exist in the design. This is a mandatory option.

-name *rule_name*

Specifies the name of the exception floorplan rule. This is a mandatory option.

-from_object_types *type_list*

Specifies the list of "from" object types for the exception floorplan rule. These type of objects in the "from" objects of specified list of rules will be exempted from those rule's checking. Valid values for this option are *block_boundary*, *placement_blockage*, *routing_blockage*, *hard_macro*, *soft_macro* and *std_cell_area*. This option is mutually exclusive with *-object_types*, *-from_lib_cells* and *-lib_cells*.

-from_lib_cells *lib_cells*

Specifies the collection of "from" lib cells for the exception floorplan rule. These lib cells in the "from" lib cells of specified list of rules will be exempted from those rule's checking. This option is mutually exclusive with *-from_object_types*, *-object_types*, and *-lib_cells*.

-to_object_types *type_list*

Specifies the list of "to" object types for the exception floorplan rule. These type of objects in the "to" objects of specified list of rules will be exempted from those rule's checking. Valid values for this option are *block_boundary*, *placement_blockage*, *routing_blockage*, *hard_macro*, *soft_macro* and *std_cell_area*. This option is mutually exclusive with *-object_types*, *-to_lib_cells* and *-lib_cells*.

-to_lib_cells *lib_cells*

Specifies the collection of "to" lib cells for the exception floorplan rule. These lib cells in the "to" lib cells of specified list of rules will be exempted from those rule's checking. This option is mutually exclusive with *-to_object_types*, *-object_types*, and *-lib_cells*.

-object_types *type_list*

Specifies the list of both object types for the exception floorplan rule. These type of objects in any objects of specified list of rules will be exempted from those rule's checking. Valid values for this option are *block_boundary*, *placement_blockage*, *routing_blockage*, *hard_macro*, *soft_macro* and *std_cell_area*. This option is mutually exclusive with *-from_object_types*, *-to_object_types*, *-from_lib_cells*, *-to_lib_cells* and *-lib_cells*.

-lib_cells *lib_cells*

Specifies the collection of both lib cells for the exception floorplan rule. These lib cells in any lib cells of specified list of rules will be exempted from those rule's checking. This option is mutually exclusive with *-from_object_types*, *-to_object_types*, *-object_types*, *-from_lib_cells* and *-to_lib_cells*.

-identical

Specifies whether this rule applies to hard macros of same reference.

-orientation_types *orientation_list*

Specifies the orientation of the two objects for the check to be enabled. Valid values are *align*, *mirror* and *partial*. *align* means both the objects should be of same orientation like R0, MX, MY or R180. *partial* means the orientation pair should be R0-R180 or MX-MY. *mirror* means the objects are mirrored in checked direction.

-reason *reason*

Specifies the reason for this exception rule to be specified.

DESCRIPTION

The **set_floorplan_exception_rules** command defines a named exception floorplan rule in the current design. The defined rule is persistent. If another floorplan rule by the same name exists then the command errors out.

The exception rule is to apply existing rules on a subset of objects or lib cells specified for those rules. If a spacing rule has all hard macro and soft macro and if user would like to exclude certain lib cells from that list then an exception rule can be created.

EXAMPLES

The following example creates an exception rule named exc1 to exempt checking of "from" std_cell_area and "to" an21* lib cells in the spacing floorplan rule s1 and enclosure floorplan rule e2.

```
prompt> set_floorplan_exception_rules -name exc1 -rules {s1 e2} \
  -from_object_types std_cell_area -to_lib_cells */an21* \
  -reason myReason
```

SEE ALSO

set_floorplan_area_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_extension_rules(2)
set_floorplan_forbidden_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_length_rules(2)
set_floorplan_spacing_rules(2)
set_floorplan_width_rules(2)
remove_floorplan_rules(2)
report_floorplan_rules(2)

set_floorplan_extension_rules

Defines an extension floorplan rule in the design.

SYNTAX

set_floorplan_extension_rules
-rules *rule_list*
-name *rule_name*
-sides *side_list*
-target *target_type*
-value *distance*

Data Types

rule_list list
rule_name string
side_list list
target_type string
distance float

ARGUMENTS

-rules *rule_list*

Specifies the list of floorplan rules for which this extension rule needs to be applied. It is not required that rules specified in this list has to pre-exist in the design. This is a mandatory option.

-name *rule_name*

Specifies the name of the extension floorplan rule. This is a mandatory option.

-sides *side_list*

Specifies the sides or directions in which the target object in specified floorplan rules is to be extended. Valid values are *all*, *bottom*, *horizontal*, *left*, *right*, *top* and *vertical*. The *horizontal* argument includes both *left* and *right*. Similarly, the *vertical* argument includes both *bottom* and *top*. This is a mandatory option.

-target *target_type*

Specifies the target object in specified floorplan rules on which this extension rule needs to be applied. It can be "from" or "to" or both objects in that specified floorplan rule. Those objects will be extended by the value specified in this extension rule. This is a mandatory option.

-value *distance*

Specifies the extent by which the target objects of specified floorplan rules will be extended for that particular rule checking. This is a mandatory option.

DESCRIPTION

The **set_floorplan_extension_rules** command defines a named extension floorplan rule in the current design. The defined rule is persistent. If another floorplan rule by the same name exists then the command errors out.

The command specifies how the target objects either in "from" or "to" or both in specified floorplan rules need to be extended during that particular rule check.

EXAMPLES

The following example creates an extension rule named ex1 to extend the "from" objects in length floorplan rules le1 and le2 in directions top and left by a value of 17 during the check for le1 and le2.

```
prompt> set_floorplan_extension_rules -name ex1 \
  -rules {le1 le2} -target from \
  -sides {top left} -value 17
```

SEE ALSO

set_floorplan_area_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_exception_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_forbidden_rules(2)
set_floorplan_length_rules(2)
set_floorplan_spacing_rules(2)
set_floorplan_width_rules(2)
remove_floorplan_rules(2)
report_floorplan_rules(2)

set_floorplan_forbidden_rules

Defines a forbidden floorplan rule in the design.

SYNTAX

set_floorplan_forbidden_rules
-lib_cells *lib_cells*
-name *rule_name*

Data Types

lib_cells collection
rule_name string

ARGUMENTS

-lib_cells *lib_cells*

Specifies the collection of library cells for the forbidden floorplan rule. Specified lib cells are not allowed to be placed in the design. This is a mandatory option.

-name *rule_name*

Specifies the name of the forbidden floorplan rule. This is a mandatory option.

DESCRIPTION

The **set_floorplan_forbidden_rules** command defines a named forbidden floorplan rule in the current design. The defined rule is persistent. If another floorplan rule by the same name exists then the command errors out.

The **set_floorplan_forbidden_rules** command is used to specify list of lib cells that are not allowed to be placed in the design.

EXAMPLES

The following example creates a forbidden rule named f1 to check that specified lib cells are not placed in the design.

```
prompt> set_floorplan_forbidden_rules -name f1 \
```

-lib_cells {*/an2* */aoi*}

SEE ALSO

set_floorplan_area_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_extension_rules(2)
set_floorplan_exception_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_length_rules(2)
set_floorplan_spacing_rules(2)
set_floorplan_width_rules(2)
remove_floorplan_rules(2)
report_floorplan_rules(2)

set_floorplan_halo_rules

Defines a halo floorplan rule in the design.

SYNTAX

set_floorplan_halo_rules
-from_object_types *from_type_list*
-to_object_types *to_type_list*
-to_lib_cells *lib_cells*
-sides *side_list*
-type inner | outer
-name *rule_name*
[-shielding_object_types *type_list*]
[-shielding_lib_cells *lib_cells*]
[-must_enclose]
[-follow_rotations]
[-ignore_rotate90]
[-layers *layer_list*]
[-to_layers *layer_list*]
[-forbidden_list *distance_list*]
[-forbidden_ranges {{*low high*} {*low1 high1*} ... }]
[-max *distance*]
[-min *distance*]
[-offset *distance*]
[-step *distance*]
[-valid_list *distance_list*]
[-valid_ranges {{*low high*} {*low1 high1*} ... }]

Data Types

from_type_list list
to_type_list list
lib_cells collection
type_list list
side_list list
rule_name string
layer_list list
distance_list list
low float
high float
low1 float
high1 float
distance float

ARGUMENTS

-from_object_types *from_type_list*

Specifies the list of "from" object types for the halo floorplan rule. These type of objects will form the halo around other objects specified with **-to_object_types** or library cells specified with **-to_lib_cells**. Valid values for this option are *core_area*, *placement_blockage*, *routing_blockage*, *shape* and *std_cell_area*. This is a mandatory option.

-to_object_types *to_type_list*

Specifies the list of "to" object types for the halo floorplan rule. These type of objects will be surrounded by other objects specified with **-from_object_types**. Valid values for this option are *hard_macro*, *placement_blockage*, *routing_blockage*, *soft_macro*, *shape* and *std_cell_area*. This option is mutually exclusive with **-to_lib_cells** and either one of them must be specified.

-to_lib_cells *lib_cells*

Specifies the collection of library cells for the halo floorplan rule. These library cells will be surrounded by other objects specified with **-from_object_types**. This option is mutually exclusive with **-to_object_types** and you must specify one or the other.

-shielding_object_types *type_list*

Specifies the collection of object types for the halo floorplan rule as shielding objects so that the rule will not be applied when these object types form a shield between "from" and "to" objects. Valid values for this option are *hard_macro* and *std_cell_area*. This is an optional option.

-shielding_lib_cells *lib_cells*

Specifies the collection of library cells for the halo floorplan rule as shielding objects so that the rule will not be applied when these lib cells form a shield between "from" and "to" objects.

-sides *side_list*

Specifies the sides or directions from which the "from" object surrounds the "to" object. Spacing is checked between the objects. Valid values are *all*, *bottom*, *horizontal*, *left*, *right*, *top* and *vertical*. The *horizontal* argument includes both *left* and *right*. Similarly, the *vertical* argument includes both *bottom* and *top*. This is a mandatory option.

-type inner | **outer**

Specifies the halo type for the distance check. **-type inner** specifies the inner region of surrounding object to surrounded object needs to be checked. **-type outer** specifies the outer region of surrounding object to surrounded object needs to be checked. This is a mandatory option.

-name *rule_name*

Specifies the name of the halo floorplan rule. This is a mandatory option.

-must_enclose

Specifies that the "from" object must completely enclose the "to" object from all sides. This is an optional option.

-follow_rotations

Specifies whether the sides specified by the **-sides** option should follow the rotations of the library cells, that is, if meaning of *horizontal* or *vertical* should change when the library cell has a 90-degree rotation. This option must be used together with **-to_lib_cells** or **-to_object_types hard_macro** or **-to_object_types soft_macro**. This is an optional option.

-ignore_rotate90

Specifies whether this rule can be ignored for library cells with a 90-degree rotation. This option must be used together with **-to_lib_cells** or **-to_object_types hard_macro** or **-to_object_types soft_macro**. This is an optional option.

-layers *layer_list*

Specifies the routing layers to be considered for the *routing_blockage* or *shape* in from object type. This option must be used together with **-from_object_types routing_blockage** or **-from_object_types shape**. This is an optional option.

-to_layers *layer_list*

Specifies the routing layers to be considered for the *routing_blockage* or *shape* in to object type. This option must be used together with **-to_object_types routing_blockage** or **-to_object_types shape**. This is an optional option.

-forbidden_list *distance_list*

Specifies a list of distances by which the "from" object cannot enclose the "to" object. This option is mutually exclusive with **-valid_list**. Values in the *distance_list* cannot be negative. This is an optional option.

-forbidden_ranges {{*low high*} {*low1 high1*} ... }

Specifies a list of distance ranges between which the "from" object cannot enclose the "to" object. The enclosing distance must not lie within any of *low* and *high* in the specified list of ranges. This option is mutually exclusive with **-valid_ranges**. Values cannot be negative. This is an optional option.

-max *distance*

Specifies the maximum distance by which the "from" object can enclose the "to" object. The distance cannot be greater than this value. The distance cannot be negative. If **-min** is also specified, this value must be greater than the min value. This is an optional option.

-min *distance*

Specifies the minimum distance by which the "from" object can enclose the "to" object. The distance cannot be less than this value. The value specified cannot be negative. If **-max** is also specified then this value must be less than the max value. This is an optional option.

-offset *distance*

Specifies a parameter in distance calculation between the "from" and "to" objects. This option must be used together with **-step**. This implies that the distance has to be an integer multiple of the **-step** value plus the **-offset** value. The value specified cannot be negative. This is an optional option.

-step *distance*

Specifies a parameter in distance calculation between the "from" and "to" objects. This option must be used together with **-offset**. This implies that the distance has to be an integer multiple of the **-step** value plus the **-offset** value. The value specified cannot be negative. This is an optional option.

-valid_list *distance_list*

Specifies a list of distance by which "from" object can surround the "to" object. This option is mutually exclusive with **-forbidden_list**. Values specified cannot be negative. This is an optional option.

-valid_ranges {{*low high*} {*low1 high1*} ... }

Specifies a list of distance ranges between which the "from" object can enclose the "to" object. The distance must lie within any of *low* and *high* in the specified list of ranges. This option is mutually exclusive with **-forbidden_ranges**. Values specified cannot be negative. This is an optional option.

DESCRIPTION

The **set_floorplan_halo_rules** command defines a named halo floorplan rule in the current design. The defined rule is persistent. If another floorplan rule by the same name exists then the command errors out.

There is a difference between the object type *core_area* and *std_cell_area*. The *core_area* object type means core boundary region without cutting out any blockages and is typically applicable for top level whereas *std_cell_area* object type means core boundary region after cutting out all blockages and is typically applicable for block level.

If a halo rule is defined for a library cell and another halo rule is defined for a hard macro then the halo rule defined for the library cell will take precedence over the halo rule defined for hard macro when checks are done for that library cell.

If the measured value falls inside valid range or is a member of the valid list then there is no violation given by **check_floorplan_rules** regardless of other constraints like min, max, etc. If this measured value is outside valid range or list then a violation is reported if other constraints are specified and they are not met or if no other constraints are specified.

EXAMPLES

The following example creates a halo rule by name h1 to check how hard macros and soft macros are surrounded by the core area. The check will be done for all sides for the inner region of core area. The distance can be one of 5, 7, 9 and 15. Also the distance must be an integer multiple of 3 plus 13.

```
prompt> set_floorplan_halo_rules -name h1 -from_object_types core_area \
  -to_object_types {hard_macro soft_macro} -must_enclose \
  -sides {all top} -type inner -valid_list {5 7 9 15} -offset 13 \
  -step 3
```

SEE ALSO

remove_floorplan_rules(2)
report_floorplan_rules(2)
set_floorplan_area_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_extension_rules(2)
set_floorplan_exception_rules(2)
set_floorplan_forbidden_rules(2)
set_floorplan_length_rules(2)
set_floorplan_spacing_rules(2)
set_floorplan_width_rules(2)

set_floorplan_length_rules

Defines a length floorplan rule in the design.

SYNTAX

set_floorplan_length_rules
-from_object_types *from_type_list*
-to_object_types *to_type_list*
-from_lib_cells *lib_cells*
-to_lib_cells *lib_cells*
-type parallel_run | projection_difference
-direction *direction_list*
-name *rule_name*
[-max_spacing *distance*]
[-min_parallel_run_length *distance*]
[-identical]
[-mirror]
[-asymmetric]
[-from_layers *layer_list*]
[-to_layers *layer_list*]
[-max *distance*]
[-min *distance*]

Data Types

from_type_list list
to_type_list list
lib_cells collection
direction_list list
rule_name string
distance float
layer_list list

ARGUMENTS

-from_object_types *from_type_list*

Specifies the list of "from" object types for the length floorplan rule. Length between these type of objects and other objects specified with **-to_object_types** or library cells specified with **-to_lib_cells** will be checked. Valid values for this option are *hard_macro*, *routing_blockage*, *shape*, *soft_macro* and *std_cell_area*. This option is mutually exclusive with **-from_lib_cells** and you must specify one or the other.

-to_object_types *to_type_list*

Specifies the list of "to" object types for the length floorplan rule. Length between these type of objects and other objects specified with **-from_object_types** or library cells specified with **-from_lib_cells** will be checked. Valid values for this option are *hard_macro*, *routing_blockage*, *shape*, *soft_macro* and *std_cell_area*. This option is mutually exclusive with **-to_lib_cells** and you must specify one or the other.

-from_lib_cells *lib_cells*

Specifies the collection of library cells for the length floorplan rule. Length between these library cells and other objects specified with **-to_object_types** or library cells specified with **-to_lib_cells** will be checked. This option is mutually exclusive with **-from_object_types** and you must specify one or the other.

-to_lib_cells *lib_cells*

Specifies the collection of library cells for the length floorplan rule. Length between these library cells and other objects specified with **-from_object_types** or library cells specified with **-from_lib_cells** will be checked. This option is mutually exclusive with **-to_object_types** and you must specify one or the other.

-type parallel_run | projection_difference

Specifies the length type of the length check. Valid values are *parallel_run* and *projection_difference*. This is a mandatory option.

-direction *direction_list*

Specifies the sides or directions in which length of object or library cells needs to be checked. Valid values are *horizontal* and *vertical*. The *horizontal* argument includes both *left* and *right*. Similarly, the *vertical* argument includes both *bottom* and *top*. This is a mandatory option.

-name *rule_name*

Specifies the name of the length floorplan rule. This is a mandatory option.

-max_spacing *distance*

Specifies the maximum spacing between "from" and "to" objects for this rule to be applied. The specified value cannot be negative. This is an optional option.

-min_parallel_run_length *distance*

Specifies the minimum overlap length between "from" and "to" objects for this rule to be applied. This is an optional option.

-identical

Specifies whether this rule applies to hard macros of same reference. This is an optional option.

-mirror

Specifies whether this rule applies when hard macros face each other mirrored. This option must be used together with **-identical**. This is an optional option.

-asymmetric

Specifies whether this rule applies asymmetric or not. If specified, need to measure the difference only from object specified in **-from_object_types**. This option must be used along with **-type projection_difference**. This is an optional option.

-from_layers *layer_list*

Specifies the routing layers to be considered for **routing_blockage** or **shape** object types. This option must be used along with **-from_object_types routing_blockage** or **-from_object_types shape**. This is an optional option.

-to_layers *layer_list*

Specifies the routing layers to be considered for **routing_blockage** or **shape** object types. This option must be used along with **-**

to_object_types routing_blockage or **-to_object_types shape**. This is an optional option.

-max ***distance***

Specifies the maximum value of length type between "from" object or "from" library cell and "to" object or "to" library cell. The distance cannot be greater than this value. The specified value cannot be negative. If **-min** is also specified then this value must be greater than the min value. This is an optional option.

-min ***distance***

Specifies the minimum value of length type between "from" object or "from" library cell and "to" object or "to" library cell. The distance cannot be less than this value. The specified value cannot be negative. If **-max** is also specified then this value must be lesser than the max value. This is an optional option.

DESCRIPTION

The **set_floorplan_length_rules** command defines a named length floorplan rule in the current design. The defined rule is persistent. If another floorplan rule by the same name exists then the command errors out.

There is a difference between the object type *core_area* and *std_cell_area*. The *core_area* object type means core boundary region without cutting out any blockages and is typically applicable for top level whereas *std_cell_area* object type means core boundary region after cutting out all blockages and is typically applicable for block level.

If a length rule is defined for a library cell and another length rule is defined for a hard macro then the length rule defined for the library cell will take precedence over the length rule defined for hard macro when checks are done for that library cell.

EXAMPLES

The following example creates a length rule named le1 to check length between the standard cell area and the hard macro in the vertical direction, both top and bottom. The length must be at least 5.

```
prompt> set_floorplan_length_rules -name le1 \
  -from_object_types std_cell_area -to_object_types hard_macro \
  -direction vertical -min 5
```

SEE ALSO

set_floorplan_area_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_extension_rules(2)
set_floorplan_exception_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_forbidden_rules(2)
set_floorplan_spacing_rules(2)
set_floorplan_width_rules(2)
remove_floorplan_rules(2)
report_floorplan_rules(2)

set_floorplan_location_rules

Defines a location floorplan rule in the design.

SYNTAX

set_floorplan_location_rules
-from_object_types *type_list*
-to_object_types *type_list*
-from_lib_cells *lib_cells*
-to_lib_cells *lib_cells*
-from_type *location_type*
-to_type *location_type*
[-follow_rotations]
[-ignore_rotate90]
-direction *direction*
[-from_layers *layer_list*]
[-to_layers *layer_list*]
-name *rule_name*
[-forbidden_list *distance_list*]
[-forbidden_ranges {{*low high*} {*low1 high1*} ... }]
[-max *distance*]
[-min *distance*]
[-offset *distance*]
[-step *distance*]
[-valid_list *distance_list*]
[-valid_ranges {{*low high*} {*low1 high1*} ... }]

Data Types

type_list list
lib_cells collection
location_type string
direction string
layer_list list
rule_name string
distance_list list
low float
high float
low1 float
high1 float
distance float

ARGUMENTS

-from_object_types *type_list*

Specifies the list of "from" object types for the location floorplan rule. Location from these type of objects for a "from" location type will be checked with other objects specified with **-to_object_types** or library cells specified with **-to_lib_cells**. Valid values for this option are *block_boundary*, *core_area*, *hard_macro*, *placement_blockage*, *routing_blockage*, *shape*, *soft_macro* and *std_cell_area*. This option is mutually exclusive with **-from_lib_cells** and you must specify one or the other.

-to_object_types *type_list*

Specifies the list of "to" object types for the location floorplan rule. Location to these type of objects for a "to" location type will be checked with other objects specified with **-from_object_types** or library cells specified with **-from_lib_cells**. Valid values for this option are *block_boundary*, *core_area*, *hard_macro*, *placement_blockage*, *routing_blockage*, *shape*, *soft_macro* and *std_cell_area*. This option is mutually exclusive with *-to_lib_cells* and you must specify one or the other.

-from_lib_cells *lib_cells*

Specifies the collection of "from" lib cells for the location floorplan rule. Location from these lib cells for a "from" location type will be checked with other objects specified with **-to_object_types** or library cells specified with **-to_lib_cells**. This option is mutually exclusive with *-from_object_types* and you must specify one or the other.

-to_lib_cells *lib_cells*

Specifies the collection of "to" lib cells for the location floorplan rule. Location to these lib cells for a "to" location type will be checked with other objects specified with **-from_object_types** or library cells specified with **-from_lib_cells**. This option is mutually exclusive with *-to_object_types* and you must specify one or the other.

-from_type *location_type*

Specifies the "from" location type of the location check. Valid values are *all_corners*, *bbox_all*, *bbox_bottom_left*, *bbox_top_right*, *bottom_left* , *bottom_right*, *top_left* and *top_right*. This is a mandatory option.

-to_type *location_type*

Specifies the "to" location type of the location check. Valid values are *all_corners*, *bbox_all*, *bbox_bottom_left*, *bbox_top_right*, *bottom_left* , *bottom_right*, *top_left* and *top_right*. This is a mandatory option.

-follow_rotations

Specifies whether mentioned *direction* should follow the rotations of library cells, that is, if meaning of *horizontal* or *vertical* should change when library cell has a 90-degree rotations. This option must be used together with **-from_lib_cells** or **-to_lib_cells** or **-to_object_types hard_macro** or **-to_object_types soft_macro** or **-from_object_types hard_macro** or **-from_object_types soft_macro**. This is an optional option.

-ignore_rotate90

Specifies whether this rule can be ignored for library cells with a 90-degree rotation. This option must be used together with **-from_lib_cells** or **-to_lib_cells** or **-to_object_types hard_macro** or **-to_object_types soft_macro** or **-from_object_types hard_macro** or **-from_object_types soft_macro**. This is an optional option.

-direction *direction*

Specifies the side or direction in which location of object or lib cells needs to be checked. Valid values are *horizontal* and *vertical*. The *horizontal* stands for both *left* and *right*. Similarly the *vertical* stands for both *bottom* and *top*. This is a mandatory option.

-from_layers *layer_list*

Specifies the routing layers to be considered for "from" routing_blockage or shape object type. This option must be used along with *-from_object_types routing_blockage* or *-from_object_types shape*. This is an optional option.

-to_layers *layer_list*

Specifies the routing layers to be considered for "to" routing_blockage or shape object type. This option must be used along with *-to_object_types routing_blockage* or *-to_object_types shape*. This is an optional option.

-name *rule_name*

Specifies the name of the location floorplan rule. This is a mandatory option.

-forbidden_list *distance_list*

Specifies a list of distance that should not be location of object or lib cell for the location type. This option is mutually exclusive with *-valid_list*. Values specified can't be negative. This is an optional option.

-forbidden_ranges {{*low high*} {*low1 high1*} ... }

Specifies a list of distance ranges that should not be location of object or lib cell for the location type. The distance must not lie within any of *low* and *high* in the specified list of ranges. This option is mutually exclusive with *-valid_ranges*. Values specified can't be negative. This is an optional option.

-max *distance*

Specifies the maximum location of object or lib cell. The distance can't be greater than this value. Value specified can't be negative. If *-min* is also specified then this value must be greater than the min value. This is an optional option.

-min *distance*

Specifies the minimum location of object or lib cell. The distance can't be lesser than this value. Value specified can't be negative. If *-max* is also specified then this value must be lesser than the max value. This is an optional option.

-offset *distance*

Specifies a parameter in location calculation of object or lib cell. This option must be used along with *-step*. This implies that the location has to be an integral multiple of *step value* plus *offset value*. Value specified can't be negative. This is an optional option.

-step *distance*

Specifies a parameter in location calculation of object or lib cell. This option must be used along with *-offset*. This implies that the location has to be an integral multiple of *step value* plus *offset value*. Value specified can't be negative. This is an optional option.

-valid_list *distance_list*

Specifies a list of distance that should be location of object or lib cell for the location type. This option is mutually exclusive with *-forbidden_list*. Values specified can't be negative. This is an optional option.

-valid_ranges {{*low high*} {*low1 high1*} ... }

Specifies a list of distance ranges that should be location of object or lib cell for the location type. The distance must lie within any of *low* and *high* in the specified list of ranges. This option is mutually exclusive with *-forbidden_ranges*. Values specified can't be negative. This is an optional option.

DESCRIPTION

The **set_floorplan_location_rules** command defines a named location floorplan rule in the current design. The defined rule is persistent. If another floorplan rule by the same name exists then the command errors out.

There is a difference between the object type *core_area* and *std_cell_area*. The *core_area* object type means core boundary region without cutting out any blockages and is typically applicable for top level whereas *std_cell_area* object type means core boundary region after cutting out all blockages and is typically applicable for block level.

If a location rule is defined for a lib cell and another location rule is defined for a hard macro then the location rule defined for the lib cell will take precedence over the location rule defined for hard macro when checks are done for that lib cell.

If the measured value falls inside valid range or is a member of the valid list then there is no violation given by **check_floorplan_rules** regardless of other constraints like min, max, etc. If this measured value is outside valid range or list then a violation is reported if other constraints are specified and they are not met or if no other constraints are specified.

EXAMPLES

The following example creates a location rule by name l1 to check the locations of a soft macro and a placement blockage such that top-right of the soft macro is vertically (both the bottom and the top) within a maximum distance of 2300 of the bottom-right of the placement blockage.

```
prompt> set_floorplan_location_rules -name l1 -from_object_types soft_macro \
  -from_type top_right -to_object_types placement_blockage -to_type bottom_right \
  -direction vertical -max 2300
```

SEE ALSO

set_floorplan_area_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_extension_rules(2)
set_floorplan_exception_rules(2)
set_floorplan_forbidden_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_length_rules(2)
set_floorplan_spacing_rules(2)
set_floorplan_width_rules(2)
remove_floorplan_rules(2)
report_floorplan_rules(2)

set_floorplan_reshape_rules

Defines a reshape floorplan rule in the design.

SYNTAX

set_floorplan_reshape_rules
-object_types *type_list*
-ignore_lib_cells *lib_cells*

Data Types

type_list	list
lib_cells	collection

ARGUMENTS

-object_types *type_list*

Specifies the list of object types for the reshape floorplan rule. Valid values for this option are *boundary_cell_region* and *std_cell_area*.

-ignore_lib_cells *lib_cells*

Specifies the collection of library cells for the reshape floorplan rule.

DESCRIPTION

The **set_floorplan_reshape_rules** command defines a named reshape floorplan rule in the current design. The defined rule is persistent. If another floorplan rule by the same name exists then the command errors out.

Reshape rule is used to connect objects underneath macro to make sure macros are enclosured with reshape object.

EXAMPLES

The following example creates a reshape rule named reshape1.

```
prompt> set_floorplan_reshape_rules -name reshape1 \
```

```
    -object_types std_cell_area \
    -ignore_lib_cells [get_lib_cells */adio*]
```

SEE ALSO

set_floorplan_area_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_extension_rules(2)
set_floorplan_exception_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_forbidden_rules(2)
set_floorplan_spacing_rules(2)
set_floorplan_width_rules(2)
remove_floorplan_rules(2)
report_floorplan_rules(2)

set_floorplan_spacing_rules

Defines a spacing floorplan rule in the design.

SYNTAX

set_floorplan_spacing_rules
-from_object_types *from_type_list*
-to_object_types *to_type_list*
-from_lib_cells *lib_cells*
-to_lib_cells *lib_cells*
-directions *direction_list*
[-orientation_types *orientation_list*]
-name *rule_name*
[-shielding_object_types *type_list*]
[-shielding_lib_cells *lib_cells*]
[-min_parallel_run_length *distance*]
[-max_parallel_run_length *distance*]
[-follow_rotations]
[-ignore_rotate90]
[-no_overlap]
[-identical]
[-mirror]
[-from_layers *from_layer_list*]
[-to_layers *to_layer_list*]
[-check_same_object]
[-forbidden_list *distance_list*]
[-forbidden_ranges {{*low high*} {*low1 high1*} ... }]
[-max *distance*]
[-min *distance*]
[-offset *distance*]
[-step *distance*]
[-valid_list *distance_list*]
[-valid_ranges {{*low high*} {*low1 high1*} ... }]

Data Types

from_type_list list
to_type_list list
lib_cells collection
orientation_list list
type_list list
direction_list list
rule_name string
distance float
from_layer_list list
to_layer_list list

distance_list list
low float
high float
low1 float
high1 float

ARGUMENTS

-from_object_types *from_type_list*

Specifies the list of "from" object types for the spacing floorplan rule. Spacing between these type of objects and other objects specified with **-to_object_types** or library cells specified with **-to_lib_cells** will be checked. Valid values for this option are *block_boundary*, *hard_macro*, *placement_blockage*, *routing_blockage*, *shape*, *soft_macro*, *unplaceable_area*, *boundary_cell_region* and *std_cell_area*. This option is mutually exclusive with **-from_lib_cells** and you must specify one or the other.

-to_object_types *to_type_list*

Specifies the list of "to" object types for the spacing floorplan rule. Spacing between these type of objects and other objects specified with **-from_object_types** or library cells specified with **-from_lib_cells** will be checked. Valid values for this option are *block_boundary*, *hard_macro*, *placement_blockage*, *routing_blockage*, *shape*, *soft_macro*, *unplaceable_area*, *boundary_cell_region* and *std_cell_area*. This option is mutually exclusive with **-to_lib_cells** and you must specify one or the other.

-from_lib_cells *lib_cells*

Specifies the collection of library cells for the spacing floorplan rule. Spacing between these library cells and other objects specified with **-to_object_types** or library cells specified with **-to_lib_cells** will be checked. This option is mutually exclusive with **-from_object_types** and you must specify one or the other.

-to_lib_cells *lib_cells*

Specifies the collection of library cells for the spacing floorplan rule. Spacing between these library cells and other objects specified with **-from_object_types** or library cells specified with **-from_lib_cells** will be checked. This option is mutually exclusive with **-to_object_types** and you must specify one or the other.

-shielding_object_types *type_list*

Specifies the collection of object types for the halo floorplan rule as shielding objects so that the rule will not be applied when these object types form a shield between "from" and "to" objects. Valid values for this option are *hard_macro* and *std_cell_area*. This is an optional option.

-shielding_lib_cells *lib_cells*

Specifies the collection of library cells for the halo floorplan rule as shielding objects so that the rule will not be applied when these lib cells form a shield between "from" and "to" objects.

-directions *direction_list*

Specifies the sides or directions in which spacing between "from" object or from library cells and "to" object or to library cells needs to be checked. Valid values are *any*, *horizontal*, *vertical*, *left*, *right*, *bottom*, *top* and *nearest_corners*. The *horizontal* argument includes both *left* and *right*. Similarly, the *vertical* argument includes both *bottom* and *top*. This is a mandatory option.

-orientation_types *orientation_list*

Specifies the orientation of the two objects for the check to be enabled. Valid values are *align*, *mirror* and *partial*. *align* means

both the objects should be of same orientation like R0, MX, MY or R180. *partial* means the orientation pair should be R0-R180 or MX-MY. *mirror* means the objects are mirrored in checked direction.

-name *rule_name*

Specifies the name of the spacing floorplan rule. This is a mandatory option.

-min_parallel_run_length *distance*

Specifies the minimum overlap length of two "to" objects or to library cells kept side-by-side. This is an optional option.

-max_parallel_run_length *distance*

Specifies the maximum overlap length of two "to" objects or to library cells kept side-by-side. This is an optional option.

-follow_rotations

Specifies whether mentioned *sides* should follow the rotations of library cells, that is, if meaning of *horizontal* or *vertical* should change when library cell has a 90-degree rotations. This option must be used together with **-to_lib_cells** or **-from_lib_cells** or **-to_object_types hard_macro** or **-to_object_types soft_macro** or **-from_object_types hard_macro** or **-from_object_types soft_macro**. This is an optional option.

-ignore_rotate90

Specifies whether this rule can be ignored for library cells with a 90-degree rotation. This option must be used together with **-to_lib_cells** or **-from_lib_cells** or **-to_object_types hard_macro** or **-to_object_types soft_macro** or **-from_object_types hard_macro** or **-from_object_types soft_macro**. This is an optional option.

-no_overlap

Specifies whether the shapes can overlap. By default the shapes can overlap. This is an optional option.

-identical

Specifies whether this rule applies to hard macros of same reference. This is an optional option.

-mirror

Specifies whether this rule applies when hard macros face each other mirrored. This option must be used together with **-identical**. This is an optional option.

-from_layers *from_layer_list*

Specifies the routing layers to be considered for **-from_object_types routing_blockage** or **-from_object_types shape**. This option must be used along with **-from_object_types routing_blockage** or **-from_object_types shape**. This is an optional option.

-to_layers *to_layer_list*

Specifies the routing layers to be considered for **-to_object_types routing_blockage** or **-to_object_types shape** object types. This option must be used along with **-to_object_types routing_blockage** or **-to_object_types shape**. This is an optional option.

-check_same_object

Specifies whether this rule will check the spacing between edges of same objects. This option must be used along with **-from_object_types std_cell_area** and **-to_object_types std_cell_area**. This is an optional option.

-forbidden_list *distance_list*

Specifies a list of distances that are not allowed between "from" objects or "from" library cell and "to" object or "to" library cell. This option is mutually exclusive with **-valid_list**. Values specified cannot be negative. This is an optional option.

-forbidden_ranges {{*low high*} {*low1 high1*} ... }

Specifies a list of distance ranges that are not allowed between "from" objects or "from" library cell and "to" object or to library cell. The distance must not lie within any of *low* and *high* in the specified list of ranges. This option is mutually exclusive with **-valid_ranges**. Values specified cannot be negative. This is an optional option.

-max *distance*

Specifies the maximum distance between "from" object or "from" library cell and "to" object or "to" library cell. The distance cannot be greater than this value. The specified value cannot be negative. If **-min** is also specified then this value must be greater than the min value. This is an optional option.

-min *distance*

Specifies the minimum distance between "from" object or "from" library cell and "to" object or "to" library cell. The distance cannot be less than this value. The specified value cannot be negative. If **-max** is also specified then this value must be lesser than the max value. This is an optional option.

-offset *distance*

Specifies a parameter in distance calculation between "from" and "to" objects. This option must be used together with **-step**. This implies that the distance has to be an integer multiple of *step value* plus *offset value*. Value specified cannot be negative. This is an optional option.

-step *distance*

Specifies a parameter in distance calculation between "from" and "to" objects. This option must be used together with **-offset**. This implies that the distance has to be an integer multiple of *step value* plus *offset value*. Value specified cannot be negative. This is an optional option.

-valid_list *distance_list*

Specifies a list of legal separation distances between the "from" object or "from" library cell and "to" object or "to" library cell. This option is mutually exclusive with **-forbidden_list**. Values specified cannot be negative. This is an optional option.

-valid_ranges {{*low high*} {*low1 high1*} ... }

Specifies a list of distance ranges between with the "from" object or "from" library cell and "to" object or "to" library cell must be separated. The distance must lie within any of *low* and *high* in the specified list of ranges. This option is mutually exclusive with **-forbidden_ranges**. Values specified cannot be negative. This is an optional option.

DESCRIPTION

The **set_floorplan_spacing_rules** command defines a named spacing floorplan rule in the current design. The defined rule is persistent. If another floorplan rule by the same name exists then the command errors out.

There is a difference between the object type *core_area* and *std_cell_area*. The *core_area* object type means core boundary region without cutting out any blockages and is typically applicable for top level whereas *std_cell_area* object type means core boundary region after cutting out all blockages and is typically applicable for block level.

If a spacing rule is defined for a library cell and another spacing rule is defined for a hard macro then the spacing rule defined for the library cell will take precedence over the spacing rule defined for hard macro when checks are done for that library cell.

If the measured value falls inside valid range or is a member of the valid list then there is no violation given by **check_floorplan_rules** regardless of other constraints like min, max, etc. If this measured value is outside valid range or list then a violation is reported if other constraints are specified and they are not met or if no other constraints are specified.

EXAMPLES

The following example creates a spacing rule named s1 to check spacing between the standard cell area and the block boundary in the vertical direction, both top and bottom. The spacing must be at least 5.

```
prompt> set_floorplan_spacing_rules -name s1 \
  -from_object_types std_cell_area -to_object_types block_boundary \
  -directions vertical -min 5
```

SEE ALSO

set_floorplan_area_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_extension_rules(2)
set_floorplan_exception_rules(2)
set_floorplan_forbidden_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_length_rules(2)
set_floorplan_width_rules(2)
remove_floorplan_rules(2)
report_floorplan_rules(2)

set_floorplan_unplaceable_area_extension_rules

Defines an unplaceable area extension floorplan rule.

SYNTAX

set_floorplan_unplaceable_area_extension_rules
-object_types *type_list*
-lib_cells *lib_cells*
-sides *side_list*
-value *distance*
-name *rule_name*
[-follow_rotations]

Data Types

type_list	list
lib_cells	collection
side_list	list
distance	float
rule_name	string

ARGUMENTS

-object_types *type_list*

Specifies the list of object types for the unplaceable area extension floorplan rule. The area of these type of objects will be checked. Valid values for this option are *std_cell_area* and *hard_macro*. This option is mutually exclusive with **-lib_cells** and you must specify one or the other.

-lib_cells *lib_cells*

Specifies the collection of library cells for the unplaceable area extension floorplan rule. This option is mutually exclusive with **-object_types** and you must specify one or the other.

-sides *side_list*

Specifies the sides or directions in which the target object in specified floorplan rules is to be extended. Valid values are *all*, *bottom*, *horizontal*, *left*, *right*, *top* and *vertical*. The *horizontal* argument includes both *left* and *right*. Similarly, the *vertical* argument includes both *bottom* and *top*. This is a mandatory option.

-value *distance*

Specifies the extent by which the target objects of specified floorplan rules will be extended for that particular rule checking. This is a mandatory option.

-name *rule_name*

Specifies the name of the unplaceable area extension floorplan rule. This is a mandatory option.

-follow_rotations

Specifies whether mentioned *sides* should follow the rotations of macro, that is, if meaning of *horizontal* or *vertical* should change when macro has a 90-degree rotations. This option must be used together with **-lib_cells** or **-object_types hard_macro**. This is an optional option.

DESCRIPTION

The **set_floorplan_unplaceable_area_extension_rules** command defines a named unplaceable area extension floorplan rule in the current design. The defined rule is persistent. If another floorplan rule by the same name exists then the command errors out.

If an unplaceable area extension rule is defined for a library cell and another unplaceable area extension rule is defined for a hard macro then the rule defined for the library cell will take precedence over the rule defined for hard macro when checks are done for that library cell.

EXAMPLES

The following example creates an unplaceable area extension rule named a1 to check the area of block boundary and hard macro. The area can't be greater than 2400 and can't lie between 370 and 790.

```
prompt> set_floorplan_area_rules -name a1 \
  -object_types hard_macro -sides {top left} \
  -value 2000
```

SEE ALSO

remove_floorplan_rules(2)
report_floorplan_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_extension_rules(2)
set_floorplan_exception_rules(2)
set_floorplan_forbidden_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_length_rules(2)
set_floorplan_spacing_rules(2)
set_floorplan_width_rules(2)

set_floorplan_width_rules

Defines a width floorplan rule.

SYNTAX

set_floorplan_width_rules
-object_types *type_list*
-lib_cells *lib_cells*
-type concave | incorner | jog | simple
-direction *direction_list*
-name *rule_name*
[-except_project_lib_cells *lib_cells*]
[-project_extension *distance*]
[-max_project_spacing *distance*]
[-layers *layer_list*]
[-forbidden_list *width_list*]
[-forbidden_ranges {{*low high*} {*low1 high1*} ... }]
[-max *distance*]
[-min *distance*]
[-offset *distance*]
[-step *distance*]
[-valid_list *distance_list*]
[-valid_ranges {{*low high*} {*low1 high1*} ... }]

Data Types

type_list list
lib_cells collection
direction_list list
rule_name string
layer_list list
width_list list
low float
high float
low1 float
high1 float
distance float
distance_list float

ARGUMENTS

-object_types *type_list*

Specifies the list of object types for the width floorplan rule. The width of these type of objects will be checked. Valid values for this option are *block_boundary*, *core_area*, *hard_macro*, *placement_blockage*, *routing_blockage*, *shape*, *soft_macro*, *unplaceable_area*, *boundary_cell_region* and *std_cell_area*. This option is mutually exclusive with **-lib_cells** and you must specify one or the other.

-lib_cells *lib_cells*

Specifies the collection of library cells for the width floorplan rule. The width of these library cells will be checked. This option is mutually exclusive with **-object_types** and you must specify one or the other.

-type concave | incorner | jog | simple

Specifies the width type of the width check. Valid values are *concave*, *incorner*, *jog* and *simple*. This is a mandatory option.

-direction *direction_list*

Specifies the sides or directions in which width of object or library cells needs to be checked. Valid values are *any*, *horizontal* and *vertical*. The *horizontal* argument includes both *left* and *right*. Similarly, the *vertical* argument includes both *bottom* and *top*. This is a mandatory option.

-name *rule_name*

Specifies the name of the width floorplan rule. This is a mandatory option.

-except_project_lib_cells *lib_cells*

Specifies the collection of library cells for the width floorplan rule where the rule will not be applied on the object if specified lib cells project on that object for a given projection distance. This is an optional option. This option is mutually exclusive with **-project_lib_cells** and you must specify one or the other.

-project_extension *distance*

Specifies the projection distance of lib cells specified with **-except_project_lib_cells** on the object so that this rule will not be applied for that object. The value specified cannot be negative. This option must be used together with **-except_project_lib_cells**. This is an optional option.

-max_project_spacing *distance*

Specifies the maximum spacing of lib cells specified with **-except_project_lib_cells** to the object within which this rule will not be applied for that object. The value specified cannot be negative. This option must be used together with **-except_project_lib_cells**. This is an optional option.

-layers *layer_list*

Specifies the routing layers to be considered for the *routing_blockage* or *shape* object type. This option must be used together with **-object_types routing_blockage** or **-object_types shape**. This is an optional option.

-project_lib_cells *lib_cells*

Specifies the collection of library cells for the width floorplan rule where the rule will be applied on the object if specified lib cells project on that object for a given projection distance. This option must be used together with **-object_types std_cell_area** and **-type simple**. This option is mutually exclusive with **-except_project_lib_cells** and you must specify one or the other.

-forbidden_list *width_list*

Specifies a list of widths that are not allowed for the object or library cell. This option is mutually exclusive with **-valid_list**. Values specified cannot be negative. This is an optional option.

-forbidden_ranges {{*low high*} {*low1 high1*} ... }

Specifies a list of width ranges that are not allowed for the object or library cell. The width must not lie within any of *low* and *high* in the specified list of ranges. This option is mutually exclusive with **-valid_ranges**. Values specified cannot be negative. This is

an optional option.

-max *distance*

Specifies the maximum width for the object or library cell. The width cannot be greater than this value. The value specified cannot be negative. If **-min** is also specified then this value must be greater than the min value. This is an optional option.

-min *distance*

Specifies the minimum width of object or library cell. The width cannot be less than this value. The value specified cannot be negative. If **-max** is also specified then this value must be lesser than the max value. This is an optional option.

-offset *distance*

Specifies a parameter in width calculation of object or library cell. This option must be used together with **-step**. This implies that the width has to be an integer multiple of *step value* plus *offset value*. Value specified cannot be negative. This is an optional option.

-step *distance*

Specifies a parameter in width calculation of object or library cell. This option must be used together with **-offset**. This implies that the width has to be an integer multiple of *step value* plus *offset value*. Value specified cannot be negative. This is an optional option.

-valid_list *distance_list*

Specifies a list of allowed widths for the width of the object or library cell. This option is mutually exclusive with **-forbidden_list**. Values specified cannot be negative. This is an optional option.

-valid_ranges {{*low high*} {*low1 high1*} ... }

Specifies a list of allowed width ranges for the object or library cell. The distance must lie within any of *low* and *high* in the specified list of ranges. This option is mutually exclusive with **-forbidden_ranges**. Values specified cannot be negative. This is an optional option.

DESCRIPTION

The **set_floorplan_width_rules** command defines a named width floorplan rule in the current design. The defined rule is persistent. If another floorplan rule by the same name exists then the command errors out.

There is a difference between the object type *core_area* and *std_cell_area*. The *core_area* object type means core boundary region without cutting out any blockages and is typically applicable for top level whereas *std_cell_area* object type means core boundary region after cutting out all blockages and is typically applicable for block level.

If a width rule is defined for a library cell and another width rule is defined for a hard macro then the width rule defined for the library cell will take precedence over the width rule defined for hard macro when checks are done for that library cell.

If the measured value falls inside valid range or is a member of the valid list then there is no violation given by **check_floorplan_rules** regardless of other constraints like min, max, etc. If this measured value is outside valid range or list then a violation is reported if other constraints are specified and they are not met or if no other constraints are specified.

EXAMPLES

The following example creates a width rule named w1 to check the concave width of block boundary and hard macro in horizontal direction, both left and right. The width must be at least 31 and can be at most 153. Also the distance needs to follow integral multiples of 2 plus 34.

```
prompt> set_floorplan_width_rules -name w1 \
  -object_types {block_boundary hard_macro} -type concave \
  -direction horizontal -max 153 -min 31 -offset 34 -step 2
```

SEE ALSO

remove_floorplan_rules(2)
report_floorplan_rules(2)
set_floorplan_area_rules(2)
set_floorplan_enclosure_rules(2)
set_floorplan_extension_rules(2)
set_floorplan_exception_rules(2)
set_floorplan_forbidden_rules(2)
set_floorplan_halo_rules(2)
set_floorplan_length_rules(2)
set_floorplan_spacing_rules(2)

set_freeze_ports

Sets the **freeze_ports** attribute on cells to prevent synthesis from modifying their port signature during optimization.

SYNTAX

string **set_freeze_ports** [-all|-clock|-data] *object_list* [*value*]

list *cell_list*
Boolean *value*

ARGUMENTS

-all|-clock|-data

Specifies the scope of the freeze setting and can limit it to clock ports, data ports, or both (the default).

cell_list

Specifies a list of cells on which to place the **freeze_ports** attribute.

value

Specifies the value with which to set the **freeze_ports** attribute. Allowed values are *true* (the default) or *false*.

DESCRIPTION

Sets the **freeze_ports** attribute on cells in the current design to prevent these objects from having ports added or removed during optimization. This attribute can be set separately to control clock and data port optimization.

The **freeze_clock_ports** and **freeze_data_ports** attributes on cells is entered by users.

EXAMPLES

The following commands specify not to modify the data ports of "block1" and "analog1" cells during synthesis.

```
prompt> set_freeze_ports -data [get_cells {block1 analog1}]
1
```

SEE ALSO

get_attribute(2)

set_grid

Updates a grid.

SYNTAX

collection **set_grid**
[-pg_strategy *strategy_name*]
[-layers *layers*]
[-site_rows *rows*]
[-site_arrays *site_arrays*]
[-x_step *step*]
[-y_step *step*]
[-x_offset *offset*]
[-y_offset *offset*]
[-orientations *allowed_orientations*]
[-reset]
grid

Data Types

strategy_name	string
layers	collection
rows	collection
step	float
offset	float
allowed_orientations	list_of_strings
grid	grid_object

ARGUMENTS

-pg_strategy *strategy_name*

Specifies the name of the PG strategy to be considered when updating the grid. This option is applicable only when updating a block grid. The tool derives the grid pitches and offsets from a combination of the specified PG strategy and any existing PG strategies and site rows of the grid. This is an option for auto-derived block grid. It is mutually exclusive with any option for manual block grid.

-layers *layers*

Specifies the tech layers to consider within the PG strategy. This option is applicable only when updating a block grid. A PG strategy might involve many routing layers. By default, all the layers in the specified PG strategy are used to derive the grid. If *-layers* is specified, only the specified layers in the PG strategy would be used. This is an option for auto-derived block grid. It is mutually exclusive with any option for manual block grid.

-site_rows *rows*

Specifies the site rows to be considered. This option is applicable only when updating a block grid. The tool derives the grid's pitches and offsets from the combination of the given site rows and any existing PG strategies and site rows. The specified site rows must be uniform and of the same type of unit site def. This option is used for auto-derived block grid. It is mutually exclusive with any option for manual block grid.

-site_arrays *site_arrays*

Specifies the site arrays to be considered. This option is applicable only when updating a block grid. The tool extracts the site rows from the array and then derives the grid's pitches and offsets from the combination of those site rows and any existing PG strategies and site rows. The specified site rows of site array must be uniform and of the same type of unit site def. This option is used for auto-derived block grid. It is mutually exclusive with any option for manual block grid.

-x_step *step*

Specifies the grid pitch in the x-direction. The default is the existing x_step of the grid. This option is applicable when updating either the user grid or block grid. Use this option for manual block grid. It is mutually exclusive with any option for auto-derived block grid.

-y_step *step*

Specifies the grid pitch in the y-direction. The default is the existing y_step of the grid. This option is used when updating either the user grid or block grid. This option is for manual block grid. It is mutually exclusive with any option for auto-derived block grid.

-x_offset *offset*

Specifies the grid offset to the design origin in the x-direction. The default is the existing x_offset of the grid. This option is used for updating either the user grid or the block grid. This option is for manual block grid. It is mutually exclusive with any option for auto-derived block grid.

-y_offset *offset*

Specifies the grid offset to the design origin in the y-direction. The default is the existing y_offset of the grid. This option is used for updating either the user grid or the block grid. This option is for manual block grid. It is mutually exclusive with any option for auto-derived block grid.

-orientations *allowed_orientations*

Specifies the allowed orientations for any associated block instance of the block grid being updated. This option is applicable only when updating a block grid. Specify one or more of: R0, MX, MY, R180. The default is the existing allowed orientations of the grid. This option is used for updating the manual block grid. It is mutually exclusive with any option for auto-derived block grid. The allowed orientations of an auto-derived block grid are derived automatically from the PG strategy, layer, and site rows.

-reset

Removes all existing options of the grid. If *-reset* is specified without other options, the grid is reset to the litho grid. You can either use **remove_grids** to delete the grid completely, or use **set_grid** with one or more of the above options to update the grid. This option is applicable to both user grid and block grid. If this option is specified with one or more of the above options, the tool resets the grid to litho grid and (in case of block grid) remove any existing PG strategy, layer or site rows from the grid, then apply the other options to update the grid.

grid

Specifies the grid to be updated.

DESCRIPTION

This command updates the options associated with a grid. Use this command to update both user grid and block grid.

Use the **set_block_grid_references** command to associate the block grid with a block reference design. Snapping the block instances to the block grid is done automatically inside **shape_blocks** and GUI editing with the snap setting. You can also use the **snap_cells_to_block_grid** command to move block instances to on block grid.

EXAMPLES

The following example creates an auto-derived block grid named gr1 from patterns on layers M4 and M5 of PG strategy s1. The **set_grid** command changes the grid to also include PG strategy s2.

```
prompt> create_grid -pg_strategy s1 -layers [get_layers {M4 M5}] gr1
prompt> set_grid -pg_strategy s2 gr1
```

The following example creates a manual block grid named gr2 with pitch of 20 microns in the y-direction. All associated block instances must be placed in orientation R0, or flipped around y-axis. The **set_grid** command changes the grid's pitch in both x- and y-direction to 10

```
prompt> create_grid -y_step 20 -orientations {R0 MY} gr2
prompt> set_grid -x_step 10 -y_step 10 gr2
```

The following example creates a user grid named ug1 with a pitch of 10 micron in both x- and y-directions. The **set_grid** command changes the pitch in y-direction to 20 microns.

```
prompt> create_grid -type user -x_step 10 -y_step 10 ug1
prompt> set_grid -y_step 20 ug1
```

SEE ALSO

create_grid(2)
get_grids(2)
remove_grids(2)
report_grids(2)
set_block_grid_references(2)
snap_cells_to_block_grid(2)

set_gui_stroke_binding

Set the command binding for a stroke.

SYNTAX

set_gui_stroke_binding
dictionary_name
stroke_sequence
[-builtin *builtin_cmd_name*]
[-clear]
[-tcl_cmd *tcl_command*]
[-label *tcl_command_label*]

ARGUMENTS

dictionary_name

Specify the dictionary to set the binding into. Windows that support stroke commands have one or more dictionaries to use when evaluating stroke commands.

stroke_sequence

Specifies the sequence of grids that defines the path of the stroke. See the *Stroke Sequences* section, below, for more information.

-builtin *builtin_cmd_name*

Specifies the name of a built-in command option to be executed.

-clear

Clears the existing command for the specified stroke sequence.

-tcl_cmd *tcl_command*

Specifies the Tcl command to be executed when the stroke is entered.

-label *tcl_command_label*

Specifies the menu label for the tcl command. This menu is displayed for line and snap_line strokes after a delay to provide information on the current bindings.

DESCRIPTION

Select a stroke by specifying a symbol with the mouse. The location of the down-click starts the stroke, and the path that the mouse follows while the mouse button is pressed determines the stroke that is entered. This stroke is mapped onto a numbered 3x3 grid and is transformed into a series of these grid numbers. A stroke dictionary is used to map this sequence of numbers to the command to be executed. When you release the mouse button, this command is executed.

You can use the **report_gui_stroke_builtins** command to determine the available built-in (non-Tcl) commands. You can use the **report_gui_stroke_bindings** command to determine the existing bindings.

Stroke Types

The stroke command facility supports the following modes for entering strokes: line-based, rectangle-based, and path-based. All stroke entry types generate stroke bindings that are the same.

If you are an occasional user, the line-based and rectangle-based entry modes are easier to specify, but they limit the number of strokes that can be generated. If you are an advanced user, the path-based entry mode, which is more difficult to specify, allows many more strokes to be generated.

See the man page for the **set_gui_stroke_preferences** command for complete information on stroke entry types,

Stroke Sequences

The path of a stroke is mapped onto a 3x3 grid. The size of the grid squares are determined to ensure that the stroke covers the width or height of the grid. The stroke is centered in the grid of horizontal or vertical strokes.

The grid is numbered as shown below:

1 2 3
4 5 6
7 8 9

The stroke is mapped onto the grid and converted to a sequence of these grid numbers, which is called a *stroke sequence*. For example, a diagonal stroke from the top-left corner of the grid to the bottom-right corner has a stroke sequence of **159**; a left-to-right horizontal stroke has a stroke sequence of **456**.

Since the strokes are path-based, a single grid might occur more than once in a sequence. For example, a stroke that moves horizontally left-to-right and right-to-left has a stroke sequence of **12321**.

Strokes also support the Shift, Control, and Alt modifier keys. If one or more of these modifiers are depressed when the stroke is started, they are applied to the stroke. The modifiers are prepended to the stroke sequence, with **s** delimiting Shift, **c** delimiting control, and **a** specifying Alt. The modifiers are separated from the sequence by using a colon (**:**). For example, a left-to-right horizontal stroke sequence is **123**, if unmodified; **s:123** with Shift; **c:123** with Control; and **sc:123** with Shift-Control modifiers.

A click is supported as a special (degenerate) case of a stroke. If the mouse down-click and mouse up-click event happen close enough together (in both time and location), the stroke is recognized as a click and a stroke sequence of **5** is generated.

The stroke facility also supports the definition of a default command binding that is to be used for any bindings not explicitly set. This default binding is specified by using a simple stroke sequence of **0**. If no binding is registered for a given set of modifiers and sequence, the tool searches for a stroke binding with the same modifiers and a sequence of **0** and uses that if it is found.

Built-In Commands

The application supporting stroke commands might support operations that are not available from the Tcl command language to be invoked via the stroke command. These commands are given a name, and you can use the **-builtin** option to specify a stroke binding to invoke these commands

The stroke facility always supports the built-in commands for zooming and panning listed below.

Zoom_In

Zoom in by a fixed percentage.

Zoom_Out

Zoom out by a fixed percentage

Zoom_In_Rect

Zoom in to fix the rectangle specified by the bounding box for the stroke.

Zoom_Out_Rect

Zoom out centering on the rectangle specified by the bounding box for the stroke.

Zoom_Full

Zoom and pan to fit the entire drawing centered in the window.

Pan_Center_Rect

Pan to center the rectangle specified by the bounding box for the stroke.

Tcl Command Bindings

The bindings support invoking Tcl-based commands to allow the functionality of the mechanism to be expanded. Tcl-based commands are allowed to contain key values that are substituted with data from the command before they are executed by the interpreter, thus allowing these commands to have access to the context information for the stroke. A leading percent (**%**) character specifies a key value..

The keys listed below are supported by tcl commands.

%rect

Substitutes the coordinates for the bounding box of the stroke. The value is a list of points {{*x1 y1*} {*x1 y2*}}, where you substitute bounding box coordinates for *x1*, *y1*,*x1*, and *y2*.

%startPoint

Specifies the starting point of the stroke. The value is in the form {*x y*}, where you substitute starting point coordinates for *x* and *y*.

%endPoint

Specifies the ending point of the stroke. The value is in the form {*x y*}, where you substitute starting point coordinates for *x* and *y*.

%sequence

Specifies the stroke sequence that invoked this command.

%modifiers

Specifies the modifiers of the stroke sequence that invoked this command.

EXAMPLES

The following example defines a left-to-right horizontal binding to execute the built-in operation that centers the location of the stroke in the viewport.

```
psyn_gui-t> set_gui_stroke_binding Graphics 123 \
```

-builtin Pan_Center_Rect

The following example calls a tcl command for the Shift-modified, lower-left to upper-right stroke sequence.

```
psyn_gui-t> set_gui_stroke_binding Graphics  \
-tcl s:753 my_tcl_command
```

The following example calls a tcl command taking a rectangle as an argument for the Shift-control, lower-left to upper-right stroke sequence.

```
psyn_gui-t> set_gui_stroke_binding Graphics \
ss:753 -tcl {my_tcl_command %rect}
```

SEE ALSO

report_gui_stroke_bindings(2)
report_gui_stroke_builtins(2)
set_gui_stroke_preferences(2)

set_gui_stroke_preferences

Set preferences controlling stroke command entry.

SYNTAX

set_gui_stroke_preferences
[-type *stroke_entry_type*]
[-shift]
[-ctrl]
[-alt]
[-extended_help_delay *ms_delay*]

ARGUMENTS

-type *stroke_entry_type*

Specifies the are **snap_line**, **line**, **rect**, **snap_path**, and **path**.

-shift

Sets the type for the shift modifier. This option is valid only when using the **-type** option.

-alt

Sets the type for the alt modifier. This option is valid only when using the **-type** option.

-ctrl

Sets the type for the ctrl modifier. This option is valid only when using the **-type** option.

-extended_help_delay *ms_delay*

Specifies the number of milliseconds to wait before showing the extended help (menu) for a stroke. A value of less than *500* makes the extended feedback immediate. Extremely large values suppress the extended feedback almost completely.

DESCRIPTION

Select a stroke command by specifying a symbol with the mouse. The location of the down-click starts the stroke, and the path that the mouse follows while the mouse button is pressed determines the stroke that is entered. This stroke is mapped onto a numbered 3x3 grid and is transformed into a series of these grid numbers. A stroke dictionary is used to map this sequence of numbers to the command to be executed. When you release the mouse button, this command is executed.

Use this command to control the user interface behavior when you are specifying a stroke. The preferences allow the strokes to be

used by both application experts and occasional users of the application.

The two primary types of stroke entry mechanisms are the line-based type and the stroke type. The primary preference supported is the stroke type. A line-based type determines the stroke based on the values of the stroke's start and end points. The path between the points is ignored. A line-based stroke turns the stroke facility into a form of "pie-menu" that supports nine different menu items. A path-based type determines the stroke based on the path between the stroke's start and end points. This allows an extremely large number of unique strokes to be defined, and it also requires a significantly-higher amount of precision when specifying the stroke than when using the line-based type.

The snap_line, line, and rect stroke types are all line-based strokes: the snap_path and path are path-based strokes.

Stroke Types

snap_line

The line between the start and end points is snapped onto the nearest 45 degree angle. It allows the generation of eight different stroke sequences and the click stroke sequence.

line

The line between the start and end points specifies the stroke. It allows the generation of eight different stroke sequences and the click stroke sequence.

rect

The rectangle between the start and end points specifies the stroke. It allows the generation of four different stroke sequences and the click stroke sequence.

snap_path

This path-based stroke snaps the segments of the path onto the nearest 45 degree angle to make it easier to generate stroke sequences with diagonals and straight lines. Since it is a path-based stroke, the number of unique sequences supported for this entry type is extremely large.

path

This path-based stroke is a free-formed path from the start point to the end point. Since it is a path-based stroke, the number of unique sequences supported for this entry type is extremely large.

The unmodified stroke type is the default type for all modifiers when a type has not been specified for the modifier. Different stroke types can be used for each combination of modifiers by setting the stroke type for the modifier with this command.

EXAMPLES

The following example sets the stroke type to snap_line.

```
psyn_gui-t> set_gui_stroke_preferences -type \
snap_line
```

The following example sets the stroke type for Shift-Control modified strokes to snap_line.

```
psyn_gui-t> set_gui_stroke_preferences -shift  \
-ctrl -type snap_line
```

SEE ALSO

report_gui_stroke_bindings(2)
report_gui_stroke_builtins(2)
set_gui_stroke_binding(2)

set_host_options

Specify settings for multithreaded and distributed processing.

SYNTAX

status **set_host_options**
[-name *name*]
[-max_cores *core_count*]
[-num_processes *process_count*]
[-submit_command *command*]
[-submit_protocol *protocol*]
[-timeout *time_in_seconds*]
[-work_dir *path*]
[-target ICC2 | ICV | PrimeTime | StarRC | Hspice | RedHawk | ALL]
[-add_hosts]
[*host_names*]

Data Types

name string
core_count int
process_count int
command string
protocol string
time_in_seconds int
path string
host_names list

ARGUMENTS

-name *name*

Specifies a name for this collection of options. The name can later be specified in the *remove_host_options* command to remove only these particular options.

If you do not specify the **-name** or **-target** options, the options specified in this command are set as the global options.

-max_cores *core_count*

If this option is specified by itself, it controls the maximum number of cores that can be used by multicore commands in the current (main) job.

If it is specified as part of a host or pool specification, it controls the number of cores each distributed slave job can use.

In both cases, the value is a maximum; some commands might use fewer cores than the specified value, but in no case would

they use more.

-num_processes *process_count*

Controls the level of parallelism used for job submission.

If you are defining an LSF, GRD, or custom pool, this is the maximum total number of jobs that can be submitted to it. If you are defining a collection of named hosts, this is the maximum number of jobs per host that can be started on those hosts.

You can restrict the number of active jobs to a specified number based on hardware or license resource restrictions.

The default is 1.

-submit_command *command*

Specifies the full path name of the LSF, GRD, or custom job submission program along with any desired options.

If you are defining a simple pool of machines and do not specify this option, **rsh** is used as the submission program.

-submit_protocol *protocol*

Specifies the protocol over which to submit jobs. If not specified, the tool will choose what protocol to use.

Values specified for this option are standardized to all lowercase.

-timeout *time_in_seconds*

Specifies the job submission timeout value in seconds.

If the submitted jobs do not become running jobs within the specified time, they are killed. This is to prevent hanging jobs in the queuing system.

If you do not specify this option, the system waits as long as needed for jobs to start.

Note that this option applies only to rsh and ssh. It does not apply to LSF or GRD queues.

-work_dir *path*

Specifies the directory for distributed jobs to use for run files.

The default is "./work_dir".

-target ICC2 | ICV | PrimeTime | StarRC | Hspice | RedHawk | ALL

Specifies which tool the configuration applies to.

The target can later be specified in the *remove_host_options* command to remove only options related to that target.

If you do not specify this option, the configuration applies to all tools.

-add_hosts

Specifies the host option settings to be stored for child jobs of the target tool.

This option must be used with the **-target** option.

host_names

Specifies a list of hosts that are used to run distributed jobs.

DESCRIPTION

Some products support commands that can be run in distributed mode, in which a task is broken up into many pieces and each subproblem is run independently. The **set_host_options** command defines the parameters for running a command in distributed mode.

It allows for simple setup for the new user and also for arbitrarily complex setup for the user with more sophisticated needs.

In its simplest form, the command is used to control the number of cores used by multicore commands. A multicore command can use fewer cores than this maximum, but the command can never exceed the stated maximum.

The command is versatile enough to interface to LSF, GRD, or a custom job-submission system, and has a mode in which a listed set of machines are used with the rsh command to execute remote jobs without having a queuing system at all.

If you are using a queuing system such as LSF, GRD, or a custom queuing system, you use the **set_host_options** command to specify a path to the batch submission system along with a queuing-system-specific set of command-line options that are passed along to the queuing system. These are typically used to ask for specific resources or otherwise control the queuing system. You can also control the submit timeout to make sure that jobs don't get stuck forever in a queuing system.

If you are not using a queuing system, you can still use distributed processing by providing a list of host machines, which the tool then uses to launch remote jobs with the rsh command.

You can name the collection of options that the **set_host_options** command creates by using the **-name** option. You can use the name with the **remove_host_options** command to remove only the options of that name. If you do not specify a name, the tool sets the global options as specified.

You can store the options that the **set_host_options** command creates for any tool by using the **-target** option. You can use the target with the **remove_host_options** command to remove only the options of that target. If you do not specify the target, the configurations are applied to all tools.

You can use the **report_host_options** command to report information about the defined host option sets.

There is a great deal of power and flexibility in the **set_host_options** command. You should start by using a single set of options until you fully understand how host option sets are made and how they are used by distributed commands. The EXAMPLES section below provides examples growing in complexity from basic to complex.

EXAMPLES

The following example sets up multicore operation within the parent process.

```
prompt> set_host_options -max_cores 4
  ...
## The following command uses a maximum of 4 threads.
prompt> some_threaded_command ...
```

The following distributed job example shows the use of host mode.

```
prompt> set_host_options -num_processes 2 -max_cores 4 {sleepy doc}
```

Note that in the previous example there are a total of 8 cores available (4 each from sleepy and doc). This example is for the "global" mode (unnamed host option).

When using a named host option, together with a command that takes a -host_options parameter, the behavior is different (to be consistent with the other named-host-option behavior), as max_cores settings are then applied to the distributed tasks to control multi-threading (if applicable), and the total number of distributed tasks / licensing is controlled by the -num_processes parameter.

An equivalent example for a named host option:

```
prompt> set_host_options -name rshOpt -submit_command "rsh" -num_processes 8 {sleepy doc}
```

This example allows up to 8 distributed tasks total. The tasks are allocated in the order the hosts respond, and could be up to 8 on just one host - especially if localhost is one of the choices, as localhost usually responds faster than other hosts. To limit tasks on a per-host basis, the following syntax can be used:

```
prompt> set_host_options -name rshOpt -submit_command "rsh" \
   -num_processes 8 {sleepy:5 doc:4}
```

This limits overall tasks to 8, and restricts host sleepy to 5 tasks and host doc to 4 tasks maximum.

The following example shows the use of an LSF farm. It sets the maximum number of jobs that can submitted to 12, chooses the linux64 queue, and names this set of options lsfOptions. This name can be used later to identify or delete this set of options.

```
prompt> set_host_options -num_processes 12 \
  -name "lsfOptions" \
  -submit_command "/lsf/bin/bsub -q linux64"
  ...
```

The following example shows the use of a non-standard pool. It sets the maximum number of jobs that can submitted to 8, defines the custom script for job submission, and names this set of options customOptions. This name can be used later to identify or delete this set of options.

```
prompt> set_host_options -num_processes 8 -name "customOptions" \
   -submit_command /usr/local/bin/my_submit_command
   ...
```

The following example shows the use of storing the host option settings for child jobs of a target tool.

```
prompt> set_host_options -num_processes 8 -target PrimeTime \
   -submit_command /usr/local/bin/my_submit_command
prompt> set_host_options -num_processes 4 -target PrimeTime \
   -submit_command /usr/local/bin/my_submit_command2 -add_hosts
   ...
```

SEE ALSO

remove_host_options(2)
report_host_options(2)

set_hpc_options

Apply HPC (High Performance Core) specific settings to current design

SYNTAX

status **set_hpc_options**
-list
-core core_name
-stage stage_name

Data Types

core_name string
stage_name string

ARGUMENTS

-list

List supported core names and stage names

-core core_name

Specifies the HPC core, use -list to see supported core names

-stage stage_name

Specifies the stage name, use -list to see supported stage names

DESCRIPTION

This command will apply HPC (High Performance Core) specific settings to current design. It sets the parameters and app options for synthesis, placement and route engines for better performance and power. Once the command is executed, all the controls will be applied to the design, the "hpc_core" attribute will be set to the design as well.

For advanced node such as 7nm, user needs to use set_technology command before invoking set_hpc_options.

EXAMPLES

In following example, list cores and stages supported in the tool:

```
prompt> set_hpc_options -list
cores: A53 A55 A72 A73 A75 A76
stages: place_opt clock_opt_cts clock_opt_opto route_auto route_opt
```

In following example, we apply HPC settings for a 7nm A76 core.

```
prompt> set_technology -node 7
prompt> set_hpc_options -core A76 -stage place_opt
prompt> place_opt
prompt> set_hpc_options -core A76 -stage clock_opt_cts
prompt> clock_opt -from build_clock -to route_clock
prompt> set_hpc_options -core A76 -stage clock_opt_opto
prompt> clock_opt -from final_opto
prompt> set_hpc_options -core A76 -stage route_auto
prompt> route_auto
prompt> set_hpc_options -core A76 -stage route_opt
prompt> route_opt
```

SEE ALSO

set_technology(2)

set_ideal_latency

Specifies ideal latency values for the pins in an ideal network.

SYNTAX

int **set_ideal_latency**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-rise] [-fall]
[-min] [-max]
value
object_list

float *value*
list *object_list*

ARGUMENTS

-modes *mode_list*

Specifies the modes for which to apply the latency value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is annotated separately. If this option is not given, the command applies latency for every scenario that is active with the specified corner(s). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to apply the latency value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is annotated separately. If this option is not given, the command applies latency for every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios for which to apply the latency value. Each of the specified scenarios is annotated separately. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command applies to the current scenario only.

-rise

Indicates that the *value* option represents the rise latency time. If you do not specify the **-rise** or **-fall** options, both values are set.

-fall

Indicates that the *value* option represents the fall latency time. If you do not specify the **-rise** or **-fall** option, both values are set.

-min

Indicates that the *value* option represents the minimum latency time. If you do not specify the **-min** or **-max** option, both values are set.

-max

Indicates that the *value* option represents the maximum latency time. If you do not specify the **-min** or **-max** option, both values are set.

value

Specifies an ideal latency value on leaf cell pins or top-level ports in an ideal network.

object_list

Specifies a list of leaf cell pins and top-level ports on which ideal latency is set.

DESCRIPTION

Sets an ideal latency value on leaf cell pins and top-level ports of an ideal network.

The tool uses ideal timing for ideal networks, which means that pins and ports have a specified ideal latency (from the **set_ideal_latency** command) or zero ideal latency by default. Ideal networks are normally used during pre-layout to avoid unnecessary DRCs. The specified ideal latency value provides an estimate of the latency time in the ideal network for pre-layout.

The specified latency value overrides the internally-estimated cell and net delay value and any other delay annotations. If the specified pins do not belong to an ideal network, a warning message is generated by the **report_ideal_network** command and the ideal latency *value* is ignored for those pins. Ideal latency values do not have any effect in ideal clock networks (for example, non-propagated clock networks).

The **set_ideal_latency** command affects all pins in the transitive fanout of the pins or ports on which ideal latency is specified. The total ideal latency at an ideal boundary pin is the sum of latency on the ideal network source and all ideal latencies on the path.

You can use the **set_ideal_latency** command for pins at lower levels of the design hierarchy.

To list ideal latency values, use the **report_ideal_network** command.

To remove the ideal latency values from a design, use the **remove_ideal_latency** or **reset_design** command.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example specifies a rise latency of *1.2* and a fall latency of *0.9* for ports named *A*, *B*, and *C*.

```
prompt> set_ideal_latency 1.2 -rise {A B C}
prompt> set_ideal_latency 0.9 -fall {A B C}
```

SEE ALSO

remove_ideal_latency(2)
remove_ideal_network(2)
remove_ideal_transition(2)
report_ideal_network(2)
set_ideal_transition(2)
set_ideal_network(2)

set_ideal_network

Marks a set of ports or pins in the design as sources of an ideal network. This disables timing update of cells and nets in the transitive fanout of the specified objects.

SYNTAX

status **set_ideal_network**
[-no_propagate]
object_list

Data Types

object_list list

ARGUMENTS

-no_propagate

Indicates that the ideal network is not propagated through leaf cells. Ideal properties are enabled on all nets that are electrically connected to the ideal network sources.

object_list

Specifies the objects to mark as the sources of an ideal network. Ideal network source objects can be ports or pins of leaf cells at any hierarchical level of the design.

If you specify nets in the *object_list* argument, the global driver pins of the specified nets are marked as ideal network sources. You can specify nets only when you specify the **-no_propagate** option.

DESCRIPTION

This command marks a set of ports or pins in the design as sources of an ideal network. You specify only the source of the network. The tool marks all nets, cells, and pins in the transitive fanout of ideal sources as ideal. The ideal property is automatically spread by the tool and respread as necessary after the netlist changes. The criteria for propagating the ideal property, starting at the source pins and ports, are as follows:

- A pin is marked as ideal if you specify it by using the **set_ideal_network** command, if it is a driver pin whose cell is ideal, or if it is a load pin attached to a net that is ideal.
- A net is marked as ideal if all its driving pins are ideal.

- A combinational cell is marked as ideal if at least one of its input pins is ideal and all the other input pins are either ideal, unconnected, or attached to a logic constant net. Objects with case analysis set are not treated as constant.

Propagation traverses through combinational cells but stops at sequential cells.

Design rule checks (DRCs) are disabled on ideal network objects.

Ideal networks can be specified in the clock or data network.

By default, the latency and transition times of an ideal network are zero. To override them, use the **set_ideal_latency** and **set_ideal_transition** commands. Ideal timing values do not have any effect in ideal clock networks (for example, nonpropagated clock networks).

To reverse the effect of the **set_ideal_network** command, use the **remove_ideal_network** command and specify the source pins or ports for the network. The **reset_design** command removes all ideal networks.

EXAMPLES

The following example creates an ideal network on ports in a design named TOP.

```
prompt> current_design {TOP}
prompt> set_ideal_network {IN1 IN2}
```

SEE ALSO

remove_ideal_network(2)
reset_design(2)
set_ideal_latency(2)
set_ideal_transition(2)

set_ideal_transition

Specifies ideal transition values for the pins in an ideal network.

SYNTAX

int **set_ideal_transition**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-rise] [-fall]
[-min] [-max]
value
object_list

float *value*
list *object_list*

ARGUMENTS

-modes *mode_list*

Specifies the modes for which to apply the transition value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is annotated separately. If this option is not given, the command applies transition for every scenario that is active with the specified corner(s). It is an error to give this option with the **-scenarios** option.

-corners *corner_list*

Specifies the corners for which to apply the transition value. Each active scenario of the modes specified by the **-modes** option and the corners specified by the **-corners** option is annotated separately. If this option is not given, the command applies transition for every scenario that is active with the specified mode(s). It is an error to give this option with the **-scenarios** option.

-scenarios *scenario_list*

Specifies the scenarios for which to apply the transition value. Each of the specified scenarios is annotated separately. It is an error to give this option with the **-modes** or the **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the command applies to the current scenario only.

-rise

Indicates that *value* represents the rise transition time. If you do not specify **-rise** or **-fall**, both values are set.

-fall

Indicates that *value* represents the fall transition time. If you do not specify **-rise** or **-fall**, both values are set.

-min

Indicates that *value* represents the minimum transition time. If you do not specify **-min** or **-max**, both values are set.

-max

Indicates that *value* represents the maximum transition time. If you do not specify **-min** or **-max**, both values are set.

value

Specifies an ideal transition value on leaf cell pins or top-level ports in an ideal network.

object_list

Specifies a list of leaf cell pins and top-level ports on which ideal transition is set.

DESCRIPTION

Sets an ideal transition value on leaf cell pins and top-level ports of an ideal network.

The tool uses ideal timing for ideal networks, which means that pins and ports have a specified ideal transition (from the **set_ideal_transition** command) or zero ideal transition by default. Ideal networks are normally used during pre-layout to avoid unnecessary DRCs. The specified ideal transition value provides an estimate of the transition time in the ideal network for pre-layout.

The specified transition value overrides the internally-estimated cell and net transition value and any other transition annotations. If the specified pins do not belong to an ideal network, a warning message is generated by the **report_ideal_network** command and the ideal transition *value* is ignored for those pins. Ideal transition values do not have any effect in ideal clock networks (i.e. non-propagated clock networks).

You can use **set_ideal_transition** for pins at lower levels of the design hierarchy.

To list ideal transition values, use the **report_ideal_network** command.

To remove the ideal transition values from a design, use the **remove_ideal_transition** or **reset_design** command.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example specifies a rise transition of *1.2* and a fall transition of *0.9* for ports named *A*, *B*, and *C*.

```
prompt> set_ideal_transition 1.2 -rise {A B C}
prompt> set_ideal_transition 0.9 -fall {A B C}
```

SEE ALSO

remove_ideal_latency(2)
remove_ideal_network(2)
remove_ideal_transition(2)
report_ideal_network(2)
set_ideal_latency(2)
set_ideal_network(2)

set_ieee_1500_configuration

Sets the IEEE 1500 insertion configuration for the current design.

SYNTAX

integer **set_ieee_1500_configuration**
[-wir_width *integer*]
[-wby_width *integer*]

ARGUMENTS

-wir_width *integer*

Specifies the width of the IEEE 1500 wrapper instruction register (WIR) to be implemented by IEEE 1500 test-mode control insertion.

By default, the WIR width is automatically determined by the tool based on the number of test-mode encodings needed and the type of encoding specified by the **-mode_decoding_style** option of the **set_dft_configuration** command.

-wby_width *integer*

Specifies the width of the IEEE 1500 bypass register (WBY) to be implemented by IEEE 1500 test-mode control insertion.

By default, the WBY width is 1.

DESCRIPTION

This command specifies the configuration of the IEEE 1500 test-mode control logic to be inserted in the current design. Note that IEEE 1500 logic is implemented only when IEEE 1500 test-mode control is enabled by the **set_dft_configuration -ieee_1500 enable** command.

To configure IEEE 1500 control data registers, use the **set_scan_path** command.

EXAMPLES

The following command changes the WIR width to 4:

prompt> **set_ieee_1500_configuration -wir_width 4**

SEE ALSO

set_scan_path(2)

set_ignored_layers

Sets ignored routing layers for RC estimation and congestion analysis. This command can also set the minimum and maximum routing layers for the design.

SYNTAX

status **set_ignored_layers**
[-rc_congestion_ignored_layers *layer_names*]
[-min_routing_layer *min_name*]
[-max_routing_layer *max_name*]
[-verbose]

Data Types

layer_names list
min_name string
max_name string

ARGUMENTS

-rc_congestion_ignored_layers *layer_names*

Lists the routing layers to be ignored in congestion analysis and RC estimation. This option excludes the specified layers from both RC estimation and congestion analysis. The option does not affect the behavior of the router.

-min_routing_layer *min_name*

Specifies the minimum routing layer for the design. The router respects this setting. To ensure that RC estimation and congestion analysis stay synchronized with the router, when you set this option, the tool marks all routing layers below the specified layer as fully blocked for RC and congestion estimation.

-max_routing_layer *max_name*

Specifies the maximum routing layer for the design. The router respects this setting. To ensure that RC estimation and congestion analysis stay synchronized with the router, when you set this option, the tool marks all routing layers above the specified layer as fully blocked for RC and congestion estimation.

-verbose

Displays verbose ignored layers information.

DESCRIPTION

This command sets the routing layers that are ignored during congestion analysis and RC estimation. You can use this command when you do not want to route nets on certain routing layers.

If you set the minimum and maximum routing layers for a design by using this command, the router respects these settings. If you set the minimum routing layer for the design to a certain layer, this command automatically sets all the routing layers below this layer as fully blocked layers. The same behavior also applies to the maximum routing layer setting.

A layer can be set as either ignored layer or minimum/maximum routing layer, congestion and rc value impact are similar but not identical. Minimum/maximum routing layer means the layer is fully blocked, ignored layer means not to use this layer for rc estimation and doesn't change the layer blockage directly.

The minimum and maximum routing layer settings work in conjunction with other tool controls. You can adjust the hardness of these constraints by setting the following application options:

- **route.common.net_min_layer_mode**
- **route.common.net_max_layer_mode**
- **route.common.number_of_vias_over_net_min_layer**
- **route.common.number_of_vias_over_net_max_layer**

For more information about these options, see the man pages.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example assigns the m5 and m6 layers to be ignored during congestion analysis and RC estimation:

```
prompt> set_ignored_layers -rc_congestion_ignored_layers {m5 m6}
```

The following example sets m2 as the minimum routing layer and m5 as the maximum routing layer. All layers below m2 and above m5 are set as ignored layers.

```
prompt> set_ignored_layers -min_routing_layer m2 -max_routing_layer m5
```

SEE ALSO

remove_ignored_layers(2)
report_ignored_layers(2)
report_lib(2)
route.common.net_max_layer_mode(3)
route.common.net_min_layer_mode(3)
route.common.number_of_vias_over_net_max_layer(3)
route.common.number_of_vias_over_net_min_layer(3)
route.common_options(3)

set_implementation

Specifies the implementation to use for synthetic library cell instances in a design.

SYNTAX

status **set_implementation**
implementation_name
cell_list

Data Types

implementation_name	string
cell_list	list

ARGUMENTS

implementation_name

Specifies the name of the implementation used to implement the synthetic cells in the *cell_list*.

cell_list

Specifies the synthetic library cell instances to be implemented by *implementation_name*. If more than one instance name is specified, enclose the names in quotation marks ("") or in braces ({}).

DESCRIPTION

The **set_implementation** command tags specified synthetic library cell instances with the name of an implementation to use to implement them. When the **compile** command is run, the tool chooses the user-specified implementation to implement a cell instance. Otherwise, **compile** chooses the implementation it considers most appropriate for a cell instance.

The command is used to control the implementation of instantiated or inferred DesignWare components.

The **report_resources** command lists the current implementation of all synthetic library instances, and indicates whether the implementation has been locked by **set_implementation**.

Use the **report_attribute** or **get_attribute** command to check the status of **set_implementation** of given cells. The attribute name is "forced_implementation". The attribute can be cleaned by **remove_attribute** command.

EXAMPLES

The *Dw_div* (Combinational Divider) component has five implementations (*rpl*, *cla*, *cla2*, *cla3*, *mlt*). The following example directs compile to use *cla3* implementation for *DW_div* that is instantiated as *U1*:

```
prompt> set_implementation cla3 U1
```

When the DesignWare component is inferred as an RTL operator, the synethic module name must be specified as well as the implemenation name. This is because the synthetic operator could bind more than one synthetic module. The following example directs compile to use *DW_div_uns/cla2* implementation for an inferred unsigned divider as *div_3*:

```
prompt> set_implementation DW_div_uns/cla2 div_3
```

The following example removes any effects of **set_implementation** from all synthetic cells of the current module:

```
prompt> remove_attribute -name forced_implementation [get_cells *]
```

SEE ALSO

compile(2)
get_attribute(2)
report_attribute(2)
remove_attribute(2)
report_resources(2)

set_individual_pin_constraints

Sets pin constraints on individual pins or nets.

SYNTAX

status **set_individual_pin_constraints**
[-pins *pins*]
[-nets *nets*]
[-ports *ports*]
[-cells *cells*]
[-allow_feedthroughs true | false]
[-allowed_layers *metal_layers*]
[-pin_spacing *track_count*]
[-pin_spacing_distance *pin_spacing_distance*]
[-sides *side_numbers*]
[-exclude_sides *exclude_side_numbers*]
[-offset *offset_distance*]
[-width *pin_width*]
[-length *pin_length*]
[-off_edge]
[-location {*x y*}]

Data Types

pins	collection
nets	collection
ports	collection
cells	collection
metal_layers	list
track_count	integer
pin_spacing_distance	real
side_numbers	list of integers
exclude_side_numbers	list of integers
offset_distance	real
pin_width	real or list of layer-real pairs
pin_length	real or list of layer-real pairs
x	float
y	float

ARGUMENTS

-pins *pins*

Specifies the pins that receive the constraints. Specify the full pin name from top-level design.

-nets *nets*

Applies the constraints only to the nets specified in *nets*.

-ports *ports*

Specifies the ports that receive the constraints.

-cells *cells*

Associates the net-based constraints to the specified block cells. For example, you can specify a layer constraint for a particular block cell, then specify the **-nets**, **-cells**, and **-allowed_layers** option to create the constraint. Note that cells must be on the same physical hierarchy with specified nets.

-allow_feedthroughs true | false

Specifies whether feedthrough ports can be created for the specified net. When true, the global router can create feedthrough routing on a block cells for the specified nets. When a feedthrough is created on a top-level net, the net is split into a set of new top-level nets and child-level nets if feedthrough nets are created for the child nets. Directions are assigned for the new feedthrough ports, and the netlist is modified to accommodate the new ports in the block cell.

This option can only be specified together with the **-nets** option and cannot be used together with **-cells** option. To set feedthrough constraints on individual block cells for selected nets, create a pin constraints file and use the **read_pin_constraints** command.

This option can also be used to set feedthrough allowance for the push-down into a block of pre-routed nets using **push_down_objects**. That command will check this pin constraint, and if found to be set, and set to true, will create feedthroughs based upon the pre-route multiple crossing of a block's boundary.

-allowed_layers *metal_layers*

Specifies the set of metal layers allowed for pin placement. The argument is a list of metal layers.

By default, the metal layers from M2 to the maximum metal layer (specified earlier in the floorplanning flow) are allowed for pin placement. If the maximum metal layer has not been specified, the maximum available metal layer specified in the technology file is used as the default. If the maximum layer specified is beyond the maximum metal layer for the current design, then the maximum layer in the technology file is used.

-pin_spacing *spacing_number*

Specifies the minimum number of wire tracks between adjacent pins or between a pin and a preroute wire that cuts across block edge in the normal direction. Pins that are created are always snapped to wire tracks. The default pin-to-pin spacing is one wire track. The spacing number must be 0 or a positive integer.

-pin_spacing_distance *spacing_distance*

Specifies the minimum distance between adjacent pins or between a pin and a preroute wire that cuts across a block edge in the normal direction. Pins that are created are always snapped to wire tracks.

-sides *side_numbers*

Specifies one or more block sides on which the pin must be placed.

A side number is a positive integer that starts at one, and increments by one as you proceed clockwise around each edge in the shape. Given any rectilinear shape, the left-most vertical edge is the starting edge (side number 1). If there are multiple left-most edges, then the edge 1 is the lowest left-most edge. You cannot specify a value of 0 for this option.

This option is mutually exclusive with **-exclude_sides** option.

-exclude_sides *exclude_side_numbers*

Specifies the block edges on which pins cannot be placed. The *exclude_side_numbers* argument is a list of positive integers.

A side number is a positive integer that starts at one, and increments by one as you proceed clockwise around each edge in the shape. Given any rectilinear shape, the left-most vertical edge is the starting edge (side number 1). If there are multiple left-most edges, then the edge 1 is the lowest left-most edge. You cannot specify a value of 0 for this option.

This option is mutually exclusive with **-sides** option.

-offset *offset_distance*

Specifies the distance in microns between the starting or ending point of a specified edge and the pin's center location. The starting point is the location where the edge begins as you proceed clockwise around the shape. The ending point is the location where the edge end as you proceed clockwise around the shape. The starting point of edge 1 is the lowest point of the edge, the ending point of edge 1 is the highest point of the edge the starting point of edge 2 is the leftmost point of the edge, and so on. Positive value means offset distance calculates from starting point of a specified edge and the pin's center location, vice versa for negative value.

A range can also be specified for this option. Specify range start and range end values within which the pins are allowed to place. The start and end values refer to the distances as measured to the starting point of the edge in microns (or the default unit). Negative value for range refer to the distances as measured to the ending point of the edge in microns (or the default unit). For example,

-offset {10 40}

You must specify the side information together with the offset. However, offset cannot be specified with multiple sides.

-width *pin_width*

Specifies the width of the pin. The width is perpendicular to the layer's preferred routing direction. The syntax can be a single real number to indicate the width is applied to all allowed layers (referred to as common pin width). Or alternatively, the width can be specified as a list of layer-real pairs as the following (referred to as per layer pin width):

-width {{M2 0.2} {M3 0.2} {M4 0.3} {M5 0.3}}

It means that if the pin is to be placed on layer M2 or M3, its width should be 0.2 and if on M4 or M5, 0.3. On other layers, use the default width defined by the tech file.

If user first sets the pin width to be common pin width (or per layer pin width), then in a second **set_individual_pin_constraints** sets the pin width to be per layer width (or common pin width), then the width specified in the second command will override the first.

However, if user first sets the pin width to be per layer, then in a second command sets the pin width again to be per layer but with different layers or different values, then the combined per layer pin width will be the final constraints. For example, the following example:

set_individual_pin_constraints -pins {U1/P1} -width 0.3
set_individual_pin_constraints -pins {U1/P1} -width {{M2 0.3} {M3 0.3}}

The first common pin width constraint will be overridden by the second per layer pin width. The pin width should be 0.3 on layer M2 or M3, but default width on all other layers.

And for the following example:

set_individual_pin_constraints -pins {U1/P1} -width {{M3 0.4} {M4 0.4}}
set_individual_pin_constraints -pins {U1/P1} -width {{M2 0.3} {M3 0.3}}

The pin width should be 0.3 on layer M2 or M3, 0.4 on layer M4, and default width on all other layers.

-length *pin_length*

Specifies the length of the pin. The length is in parallel to the layer's preferred routing direction. The syntax can be a single real number to indicate the length is applied to all allowed layers (referred to as common pin length). Or alternatively, the length can be specified as a list of layer-real pairs as the following (referred to as per layer pin length):

-length {{M2 0.2} {M3 0.2} {M4 0.3} {M5 0.3}}

It means that if the pin is to be placed on layer M2 or M3, its length should be 0.2 and if on M4 or M5, 0.3. On unspecified layers, the pin length is derived from the pin width. Refer to the manpage of **place_pins** for details.

-off_edge

Specifies whether the pin is created on the block boundary. This option is mutually exclusive with the **-sides** and **-offset** options.

-location ***{x y}***

Specifies the x- and y-location in the top-level design where the pin should be created. This option is mutually exclusive with the **-sides** and **-offset** options. When used together with the **-off_edge** option, the pin is placed on the closest wiretrack to the specified location. Note that in this case there will be no legalization of pin placement for off edge pins, as it is meant to be a basic placement where user is expected to give the legal location. If the location constraint is a hard constraint set by the **set_block_pin_constraints** command, the center of the pin is placed at the exact location specified by this option. If you do not specify the **-off_edge** option, the pins are placed on the edge that is closest to this location.

DESCRIPTION

This command sets pin placement constraints on individual pins, nets, or ports. The constraints are honored during pin placement. The pin placement constraints are saved in the design database. Except -allow_feedthroughs option, all the other options can be specified for either pins, ports or nets.

The **set_individual_pin_constraints** command is additive. The constraint values set in previous calls are retained unless they are explicitly overridden by subsequent calls. This command is aware of multiple levels of physical design hierarchy (MPH-aware).

The **set_editability** command can enable or disable the **set_individual_pin_constraints** command for specified blocks or levels of physical hierarchy. This command has no effect on the blocks that are disabled with the **set_editability** command.

This command returns 1 on success, 0 otherwise.

Please note, starting from 2019.12 release, a new command **create_pin_constraint** is introduced that can also creates the individual pin constraints (as well as bundle pin constraints) and has similar command line options. The major difference is that **create_pin_constraint** returns a collection of constraints it created, which under certain scenarios might be desired.

EXAMPLES

The following example sets the allowed pin layers for net clk to M2 and M3.

```
prompt> set_individual_pin_constraints -allowed_layers [get_layers {M2 M3}] \
  -nets [get_nets clk]
```

SEE ALSO

place_pins(2)
push_down_objects(2)
read_pin_constraints(2)

remove_individual_pin_constraints(2)
report_individual_pin_constraints(2)
set_block_pin_constraints(2)
set_editability(2)
create_pin_constraint(2)
remove_pin_constraints(2)

set_input_delay

Defines the arrival time relative to a clock.

SYNTAX

string **set_input_delay**
[-clock *clock_name*]
[-reference_pin *pin_port_name*]
[-clock_fall]
[-level_sensitive]
[-rise]
[-fall]
[-max]
[-min]
[-add_delay]
[-network_latency_included]
[-source_latency_included]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
delay_value
port_pin_list

Data Types

clock_name list
pin_port_name list
mode_list list
corner_list list
scenario_list list
delay_value float
port_pin_list list

ARGUMENTS

-clock *clock_name*

Specifies the clock to which the specified delay is related. If you use **-clock_fall**, you must specify **-clock clock_name**. If you do not specify **-clock**, the delay is relative to time zero for combinational designs. For sequential designs, the delay is considered relative to a new clock with a period determined by considering the sequential cells in the transitive fanout of each port.

-reference_pin *pin_port_name*

Specifies the clock pin or port to which the specified delay is related. If you specify this option and if propagated clocking is being used, the delay value is related to the arrival time at the specified reference pin, which is clock source latency plus its network latency from the clock source to this reference pin. The **-network_latency_included** and **-source_latency_included** options cannot be used at the same time as the **-reference_pin** option. For an ideal clock network, its the sum of source latency and network latency applied.

The pin specified with the **-reference_pin** option should be a leaf pin or port in a clock network, in the direct or transitive fanout of a clock source specified with the **-clock** option. If multiple clocks reach the port or pin where you are setting the input delay, and if the **-clock** option is not used, the command considers all of the clocks.

-clock_fall

Specifies that the delay is relative to the falling edge of the clock. When it is used with **-reference_pin** option, the delay is relative to the falling transition of the reference pin. The default is the rising edge or rising transition of a reference pin.

-level_sensitive

Specifies that the source of the delay is a level-sensitive latch. This allows the tool to derive setup and hold relationship for paths from this port as if it were a level-sensitive latch. If you do not use **-level_sensitive**, the input delay is treated as if it were a path from a flip-flop.

-rise

Specifies that *delay_value* refers to a rising transition on specified ports of the current design. If you do not specify **-rise** or **-fall**, rising and falling delays are assumed to be equal.

-fall

Specifies that *delay_value* refers to a falling transition on specified ports of the current design. If you do not specify **-rise** or **-fall**, rising and falling delays are assumed to be equal.

-max

Specifies that *delay_value* refers to the longest path. If you do not specify **-max** or **-min**, maximum and minimum input delays are assumed to be equal.

-min

Specifies that *delay_value* refers to the shortest path. If you do not specify **-max** or **-min**, maximum and minimum input delays are assumed to be equal.

-add_delay

Specifies whether to add delay information to the existing input delay or to overwrite. Use the **-add_delay** option to capture information about multiple paths leading to an input port that are relative to different clocks or clock edges.

For example, **set_input_delay 5.0 -max -rise -clock phi1 {A}** removes all other maximum rise input delay from "A", because **-add_delay** is not specified. Other input delays with different clocks or with clock_fall are removed.

In another example, **-add_delay** is specified as **set_input_delay 5.0 -max -rise -clock phi1 -add_delay {A}**. If there is an input maximum rise delay for "A" relative to clock "phi1" rising edge, the larger value is used. The smaller value does not result in critical timing for maximum delay. For minimum delay, the smaller value is used. If there is maximum rise input delay relative to a different clock or different edge of the same clock, it remains with the new delay.

-network_latency_included

Specifies whether the clock network latency should not be added to the input delay value. If this option is not specified, the clock network latency of the related clock is added to the input delay value. It has no effect if the clock is propagated or the input delay is not specified with respect to any clock.

-source_latency_included

Specifies whether the clock source latency should not be added to the input delay value. If this option is not specified, the clock source latency of the related clock is added to the input delay value. It has no effect if the input delay is not specified with respect to any clock.

delay_value

Specifies the path delay. The *delay_value* must be in units consistent with the technology library used during analysis. The *delay_value* represents the amount of time that the signal is available after a clock edge. This usually represents a combinational path delay from the clock pin of a register.

port_pin_list

Provides a list of input port or internal pin names in the current design to which *delay_value* is assigned. If you specify more than one object, the objects are enclosed in braces ({}).

-modes ***mode_list***

Specifies the scenarios on which to set the input delay value. If this option is given, all scenarios of the given modes are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-corners ***corner_list***

Specifies the scenarios on which to set the input delay value. If this option is given, all scenarios of the given corners and the current mode is used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-scenarios ***scenario_list***

Specifies the scenarios on which to set the input delay value. The **-modes** or **-corners** option must not be given with this option.

DESCRIPTION

Sets input path delay values for the current design. Used with **set_load** and **set_driving_cell**, the input and output delays characterize the operating environment of the current design.

The **set_input_delay** command sets input path delays on input ports relative to a clock edge. Unless specified, input ports are assumed to have zero input delay. For inout (bidirectional) ports, you can specify the path delays for both input and output modes.

To describe a path delay from a level-sensitive latch, use the **-level_sensitive** option. If the latch is positive-enabled, set the input delay relative to the rising clock edge. If the latch is negative enabled, set the input delay relative to the falling clock edge. If time is borrowed at that latch, add that time borrowed to the path delay from the latch when determining input delay.

If for a specific clock has a max delay that is less than the min delay, the tool makes the max delay equal to the min delay.

The tool adds input delay to path delay for paths starting at primary inputs, and to output delay for paths ending at primary outputs.

Input delay on a clock port is treated as clock source latency if the clock does not have a clock source latency defined. This way of defining clock source latency will not be supported for future releases. If the clock port leads to a D pin of a register and the clock port has both an input_delay and a clock source latency set on it, there might be double counting for the data-path report (both the input delay value and clock source latency will be added).

Input delay is managed on a per-scenario basis. For designs with multiple scenarios, you can specify different delays for different scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the given delay is applied to the current scenario. It is an error if there is no current scenario. If the **-scenarios** option is given, the delay is applied to all of the specified scenarios. If the **-modes** option is given, the

delay is applied to all scenarios of the specified modes, and it is an error if none of the specified modes have any scenarios. If the **-corners** option is given, the delay is applied to all scenarios of the specified corners and the current mode, and it is an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are given, the delay is applied to all scenarios of the specified corners and the specified modes, and it is an error there are no such scenarios. It is an error to give **-scenarios** with **-modes** or **-corners**.

To list input delays associated with ports, use **report_ports**. To remove input delay values, use **remove_input_delay** or **reset_design**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example sets an input delay of 2.3 for ports "IN1" and "IN2" on a combinational design. Because the design is combinational, no clock is needed.

```
prompt> set_input_delay 2.3 { IN1 IN2 }
```

The following example sets input delay of 1.2 relative to the rising edge of "CLK1" for all input ports in the design.

```
prompt> set_input_delay 1.2 -clock CLK1 [all_inputs]
```

The following example sets the input and output delays for the bidirectional port "INOUT1". The input signal arrives at "INOUT1" 2.5 units after the falling edge of "CLK1". The output signal is required at "INOUT1" at 1.4 units before the rising edge of "CLK2".

```
prompt> set_input_delay 2.5 -clock CLK1 -clock_fall { INOUT1 }

prompt> set_output_delay 1.4 -clock CLK2 { INOUT1 }
```

The following example models the situation where there are three paths to input port "IN1". The first path is relative to the rising edge of "CLK1". The second path is relative to the falling edge of "CLK1". The third path is relative to the falling edge of "CLK2". The **-add_delay** option is used to indicate that new input delay information does not cause old information to be removed.

```
prompt> set_input_delay 2.2 -max clock CLK1 -add_delay { IN1 }

prompt> set_input_delay 1.7 -max clock CLK1 -clock_fall -add_delay { IN1 }

prompt> set_input_delay 4.3 -max clock CLK2 -clock_fall -add_delay { IN1 }
```

In the following example, two different maximum delays and two minimum delays for port "A" are specified using **-add_delay**. Since the information is relative to the same clock and clock edge, only the largest of the maximum values and the smallest of the minimum values are maintained (in this case, 5.0 and 1.1). If **-add_delay** is not used, the new information overwrites the old information.

```
prompt> set_input_delay 3.4 -max -clock CLK1 -add_delay { A }

prompt> set_input_delay 5.0 -max -clock CLK1 -add_delay { A }

prompt> set_input_delay 1.1 -min -clock CLK1 -add_delay { A }

prompt> set_input_delay 1.3 -min -clock CLK1 -add_delay { A }
```

SEE ALSO

all_inputs(2)
create_clock(2)
current_design(2)
get_input_delays(2)
remove_input_delay(2)
report_ports(2)
reset_design(2)
set_driving_cell(2)
set_load(2)
set_output_delay(2)

set_input_transition

Sets a fixed transition time on input or inout ports.

SYNTAX

string **set_input_transition**
[-rise]
[-fall]
[-min]
[-max]
[-clock *clock_name*]
[-clock_fall]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
transition
port_list

Data Types

clock_name string
mode_list list
corner_list list
scenario_list list
transition float
port_list list

ARGUMENTS

-rise

Sets the rise transition only.

-fall

Sets the fall transition only.

-min

Sets the transition for minimum conditions.

-max

Sets the transition for maximum conditions.

-clock *clock_name*

The input transition is set relative the specified clock.

-clock_fall

Specifies that the transition is relative to the falling edge of the clock. The default is the rising edge.

-modes *mode_list*

Specifies the scenarios on which to apply the input transition values. If this option is given, all scenarios of the specified modes are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-corners *corner_list*

Specifies the scenarios on which to apply the input transition value. If this option is given, all scenarios of the specified corners and the current mode are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-scenarios *scenario_list*

Specifies the scenarios on which to apply the input transition value. The **-modes** or **-corners** option must not be specified with this option.

transition

Specifies the port transition value. This is a floating point number greater than or equal to zero.

port_list

Specifies a list of input or inout ports.

DESCRIPTION

The **set_input_transition** command specifies a fixed transition time for a list of input or inout ports. This transition time is not affected by the capacitance of the net connected to the port. The transition time calculates delays for nets and cells in the transitive fanout of the port. The port itself has no cell delay.

The transition time can be relative to a clock by using **-clock** option. This means the transition apply only those external paths driven by the clock. Using **-clock** or **-clock_fall** option can be meaningful only when timing_slew_propagation_mode is in worst_arrival mode.

Note that the worst of input transitions specified with this command are used for worst_slew mode, regardless of the clocks that are specified. If a clock-relative input transition is set on a port, these transitions will not be accessible with the **get_attribute** command until the **-clock** and **-clock_fall** options are supported for **set_input_transition** in SDC.

To display port transition or drive capability information, use the **report_ports -drive** command.

There are two other methods of describing port drive capability. The **set_driving_cell** command causes the port to have transition time calculated as if a specified library cell was driving the net. The **set_driving_cell** command also has a cell delay equal to the load-dependent portion of the delay driving the net of the library cell. The driving cell approach is accurate for nonlinear delay models, even if the capacitance is changed. Another method is to use **set_drive_resistance**, which models the driver as a linear resistance.

Input transition is managed on a per-scenario basis. For designs with multiple scenarios, you can specify different transitions for different scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the specified transition is applied to the current scenario. The command issues an error if there is no current scenario.

If the **-scenarios** option is specified, the transition will be applied to all of the specified scenarios. If the **-modes** option is specified, the transition is applied to all scenarios of the specified modes. The command issues an error if none of the specified modes have any scenarios.

If the **-corners** option is specified, the transition is applied to all scenarios of the specified corners and the current mode. The command issues an error if none of the specified corners have any scenarios with the current mode.

If both the **-modes** and **-corners** options are specified, the transition is applied to all scenarios of the specified corners and the specified modes. The command issues an error if there are no such scenarios. It is an error to specify **-scenarios** with **-modes** or **-corners**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

This example specifies that ports matching the pattern "DATA_IN*" have a transition time of 0.75 units.

```
prompt> set_input_transition 0.75 [get_ports DATA_IN*]
```

SEE ALSO

report_ports(2)
set_clock_transition(2)
set_drive_resistance(2)
set_driving_cell(2)

set_interfaces

Clean up the interfaces of bound designs.

SYNTAX

status **set_interfaces**
[-design *design*]
-force
reference

reference string or collection

ARGUMENTS

-design *design*

Specifies the design in which to clean up the interfaces. If no design is specified, the interfaces are cleaned up in the current design.

-force

Specifies to disconnect any nets connected to pins corresponding to any ports which may get deleted.

reference

Specifies the reference block for which the interface is to be cleaned. If more than one view of the reference block is bound, then the resulting ports will match that of the specified reference. Any view with a differing port interface will be unbound from the design.

DESCRIPTION

This command cleans up mismatched port interfaces in the current design (unless otherwise specified by -design). By default, the tool will attempt to cleanup all mismatched interfaces.

If "reference" is specified, the new port interface is matched to that of the specified reference. If the other views of the specified reference are also bound and have different ports, then those instances will be unbound. If any pin that will be deleted as a result of this command, is connected to a net, then "-force" must be used. Without "-force" the command will error out. When "-force" is used the ports and all corresponding pins on the instances will be first disconnected and then deleted.

EXAMPLES

The following example cleans up all mismatched instances

prompt> **set_interfaces**

The following example cleans up all mismatched instances bound to mid,

prompt> **set_interfaces design:mid.design**

SEE ALSO

link_block(2)

set_isolate_ports

Specifies the ports to be isolated from the internal fanouts of the driver nets.

SYNTAX

string **set_isolate_ports**
[-type *buffer or inverter*]
[-driver *cell_ref*]
[-force]
list *object_list*
boolean *value*

ARGUMENTS

-type

either buffer or inverter - by default a buffer is chosen.

-driver

driver libcell type for the isolation buffers. If the buffer libcell type is not available in the library, then this option will be ignored and the most optimal buffer libcells will be selected. It is possible that the buffers can be sized for better QoR. If the user does not want the buffers to be sized, then specify -force option.

-force

If a user libcell is specified with -driver,then -force ensures that the libcells arent sized.

object_list

A list of input or output port names in the current design that are to be isolated.

value

Specifies the value with which to set the attribute. Allowed values are *true* (the default) or *false*.

DESCRIPTION

The **set_isolate_ports** command specifies the input or output ports in the current design that are to be isolated. A buffer is inserted to isolate the input port from its fanout network, or between the output port and its driver.

Note that bidirectional ports cannot be isolated using this command. In addition, no isolation is performed on a port if the net connected to the port has a dont_touch attribute or if the port has been defined as a clock source or a power pin.

Use the **set_isolate_ports** command with false (see example below) or the reset_design command to remove a port from the list of ports to be isolated. Use the **report_isolate_ports** command to list the isolation status of these ports.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies that isolation is required on the output port qout with a buffer.

prompt> **set_isolate_ports qout**

The following example removes the isolation buffer requirement constraint on the output port qout (in effect, undoing what the previous command did).

prompt> **set_isolate_ports [get_ports qout] false**

SEE ALSO

set_isolation

Defines the UPF isolation strategy for the power domains in the design.

SYNTAX

status **set_isolation**
isolation_strategy_name
-domain *power_domain*
[-isolation_power_net *isolation_power_net*]
[-isolation_ground_net *isolation_ground_net*]
[-isolation_supply *isolation_supply_set*]
[-clamp_value 0 | 1 | Z | latch]
[-isolation_signal *isolation signal*]
[-isolation_sense low | high]
[-location self | parent | fanout]
[-applies_to inputs | outputs | both]
[-applies_to_boundary upper | lower | both]
[-source *source_supply_set_name*]
[-sink *sink_supply_set_name*]
[-diff_supply_only true | false]
[-elements *objects*]
[-exclude_elements *exclude_objects*]
[-no_isolation]
[-name_prefix *prefix*]
[-name_suffix *suffix*]
[-update]

Data Types

isolation_strategy_name string
power_domain string
isolation_power_net string
isolation_ground_net string
isolation_supply_set string
isolation_signal string
source_supply_set_name string
sink_supply_set_name string
objects list
exclude_objects list
prefix string
suffix string

ARGUMENTS

isolation_strategy_name

Specifies the name of the UPF isolation strategy. The name of the isolation strategy should be unique within the specified power domain.

-domain ***power_domain***

Specifies the name of the power domain to which this UPF isolation strategy will be applied.

-isolation_power_net ***isolation_power_net***

Specifies the isolation power net that will be created for the isolation cells, based on this UPF isolation strategy.

-isolation_ground_net ***isolation_ground_net***

Specifies the isolation ground net that will be created for the isolation cells, based on this UPF isolation strategy.

-isolation_supply ***isolation_supply_set***

Specifies the supply set whose power and ground functions are to be used as the isolation power and ground nets respectively. When isolation_supply_set specified is {}, NOR/NAND-style isolation cell will be inserted based on clamp value. Currently NAND-style isolation cell is not supported. This option is mutually exclusive with the **-isolation_power_net** and **-isolation_ground_net** options.

-applies_to inputs | **outputs** | **both**

Specifies whether the given **set_isolation** command applies to all inputs or outputs or both types of ports of the power domain. The default value for this option is **outputs** without filters. The default value is **both** with any filters(-source/-sink/-diff_supply_only).

-applies_to_boundary upper | **lower** | **both**

Specifies the domain boundary that the given **set_isolation** command applies to. The default value for this option is **both** for a domain with the lower domain-boundary design attribute on, and **upper** for domains that do not have the lower domain-boundary design attribute set or have the lower domain-boundary attribute set to off.

-source ***source_supply_set_name***

Specifies the source supply set that applies to the elements of the strategy. It filters the ports and pins receiving a net that is driven by the logic that is powered by the supply set.

-sink ***sink_supply_set_name***

Specifies the sink supply set that applies to the elements of the strategy. It filters the ports and pins driving a net that fans out to the logic that is powered by the supply set.

-diff_supply_only true | **false**

Determines the isolation behavior between the driver and the receiver supply sets. If the driver is powered by the same supply set as the receiver of the port, no isolation cell is introduced into the path. The default for this options is **false**. The three options **-source**, **-sink**, **-diff_supply_only** cannot be used together at the same time.

-elements ***objects***

Specifies the objects to which this UPF isolation strategy will be applied. The objects can be pins of the root cells of the power domain or ports of the top level design for top level power domains.

-exclude_elements ***exclude_objects***

Specifies the objects to which this UPF isolation strategy won't be applied. The objects can be pins of the root cells of the power domain or ports of the top level design for top level power domains.

-clamp_value 0 | 1 | z | latch

Specifies the clamp value of the isolation cells that should be created based on this strategy. The default value of this option is 0.

-isolation_signal *isolation signal*

Specifies the isolation signal to use as the control signal of the isolation cells that should be created as a result of this isolation strategy. The *isolation_signal* can be a pin, port, or net.

-isolation_sense low | high

Specifies the isolation sense of the isolation cells that should be created as a result of this isolation strategy. The default value is high.

-location self | parent | fanout

Specifies the hierarchical location of the isolation cells that should be created as a result of this isolation strategy.

A value of self specifies that the isolation cell will be put in the hierarchy of the port being isolated. A value of parent specifies that isolation cells will be added in the parent hierarchy of the isolating port's hierarchy. A value of fanout specifies that the isolation cell will be put in the load domain.

The default value is self.

-no_isolation

Specifies that the elements that are affected by this **set_isolation** command should not be isolated.

-name_prefix *prefix*

Specifies the prefix to use for the isolation cell names.

-name_suffix *suffix*

Specifies the suffix to use for the isolation cell names.

-update

Indicates that this command adds *elements* or *exclude_elements* for a previous command with the same *strategy_name* and *domain_name* and executed in the same scope.

With the **-update** option, the command only accepts **-elements** or **-excluded_elements**, and the required option *strategy_name* and **-domain**. If specified other options with **-update**, the command will error out.

DESCRIPTION

This command defines the UPF isolation strategy for the ports of the specified power domain.

If the **-elements** and **-applies_to** options are not specified, then the isolation strategy will be applied to all output ports of the power domain.

If neither **-isolation_supply**, **-isolation_power_net** nor **-isolation_ground_net** are specified, the isolation power and ground nets are automatically set to the power and ground functions of the **default_isolation** handle of the power domain, for which the strategy is being defined.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to define a UPF isolation strategy for all output ports of the specified power domain PD1. Power domain PD1 is defined on instance shutdown_inst.

```
prompt> set_isolation isolation_1 -domain PD1 \
 -isolation_power_net PN1 -isolation_ground_net GN1
```

The following example shows how to define a UPF isolation strategy for specific ports of the specified power domain:

```
prompt> set_isolation isolation_2 -domain PD1 \
      -isolation_power_net PN1 -isolation_ground_net GN1 \
 -elements shutdown_inst/special_port
```

The following example shows how to define a UPF isolation strategy using a supply set:

```
prompt> set_isolation isolation_2 -domain PD1 \
      -isolation_supply iso_supply_set \
 -elements shutdown_inst/special_port
```

The following example shows how a UPF isolation strategy can be defined without using a supply set or supply nets.

```
prompt> set_isolation isolation_2 -domain PD1 \
 -elements shutdown_inst/special_port
```

SEE ALSO

set_isolation_control(2)

set_isolation_control

Provides additional options needed for creating isolation cells. This command is needed with most **set_isolation** commands.

This command is deprecated in UPF 3.0. The **set_isolation** command now has all the options of the **set_isolation_control** command.

SYNTAX

status **set_isolation_control**
isolation_strategy_name
-domain *power_domain*
-isolation_signal *isolation signal*
[-isolation_sense low | high]
[-location self | parent | fanout]

Data Types

isolation_strategy_name string
power_domain string
isolation_signal string

ARGUMENTS

isolation_strategy_name

Specifies the name of a UPF isolation strategy previously created by the **set_isolation** command. The option settings apply to this strategy.

This argument is required.

-domain *power_domain*

Specifies the name of the power domain previously created by the **create_power_domain** command to which the strategy options apply.

This option is required.

-isolation_signal *isolation signal*

Specifies the name of the signal that enables and disables the isolation function of cells created by the isolation strategy. The *isolation_signal* argument can be a port, pin, or net.

This option is required.

-isolation_sense low | high

Specifies the sense of the isolation signal, either **low** or **high**, that enables the isolation function of cells created by the strategy.

The default is **high**.

-location self | **parent** | **fanout**

Specifies the hierarchical location for inserting isolation cells created by the strategy. Valid values are

- **self** (the default)
  Places the isolation cells inside the design being isolated.
- **parent**
  Places the isolation cells in the parent of the design being isolated.
- **fanout**
  Places the isolation cells at all fanout locations (sinks) of the port being isolated.

DESCRIPTION

This command specifies the isolation control signal for a specified isolation strategy, and optionally, the hierarchical location for inserting isolation cells on the domain boundary.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies the control signal, ENISO, that enables the isolation function for isolation cells inserted by the strategy named ISO1 in power domain PD1. A logic low on the signal enables the isolation function. The command also specifies that isolation cells are to be inserted in at the parent level, just outside the power domain boundary.

```
prompt> set_isolation_control ISO1 -domain PD1 \
  -isolation_signal ENISO -isolation_sense low \
  -location parent
```

SEE ALSO

create_power_domain(2)
set_isolation(2)

set_label_switch_list

Sets the value of the switch list for the specified design.

SYNTAX

status **set_label_switch_list**
[-reference *references*]
labels

Data Types

\reference list
labels list

ARGUMENTS

-reference *references*

Specifies the reference or references, for which to set the label switch list. If a references is not spscified, the label switch list applies to all references for which a specific label switch list has not been set.

labels

Sets the value for the designs's label switch list. Specify *labels* as a space-delimited list of view names, enclosed within curly brackets. Note that label switch lists are order dependent. Specifying an empty label switch list value clears the label switch list setting on the design.

DESCRIPTION

This command sets the label switch list of a design. A designs' label switch list is used during linking if block is the top design.

The label switch list is a precedence-ordered list of labels that is applied when binding a design. If the top design has its label switch list explicitly set, then that label switch list is used. If not, then the top design label is used.

When binding a reference, for which there is a label precedence swith list set, binding attempts to find the design of the preferred label. If not found, the binding default to the label of the top design.

EXAMPLES

The following example sets the label swith list for reference "bot":

prompt> **set_label_switch_list -reference {bot} {postlegal mapped}**

The following example clears the label switch list for reference "bot":

prompt> **set_label_switch_list -reference {bot} {}**

It is possible to set a label precedence order for all references, and a different one for some references:

prompt> **set_label_switch_list -reference {bot} {postlegal mapped}**

prompt> **set_label_switch_list {unmapped}**

SEE ALSO

get_label_switch_list(2)

set_latch_loop_breaker

Specifies transparent latch data pins to be used as loop breaker latch data pins.

SYNTAX

status **set_latch_loop_breaker**
-pin *pin_list*
[-remove]
[-avoid]

Data Types

pin_list list

ARGUMENTS

-pin *pin_list*

Specifies data pins of transparent latches to be loop breakers.

-remove

Removes the user-specified setting on pins.

-avoid

Avoids using the specified pins as loop breakers. Requests to avoid using specified pins as loop breaker pins cannot always be honored, especially if there is a loop from the pin through combinational logic to itself.

DESCRIPTION

When sequential loops of transparent latches exist in a design, the tool selects certain latch data pins for special analysis as loop breaker data pins.

Use the **set_latch_loop_breaker** command to guide the tool in selecting data pins as loop breaker data pins. By guiding the tool to select some latch data pins as loop breakers and not others, you can report the paths of interest.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example specifies that the L1/D pin is a loop breaker pin.

```
prompt> set_latch_loop_breaker -pin [get_pins L1/D]
```

set_latency_adjustment_options

Sets latency update directives for the **compute_clock_latency** command for the current mode.

SYNTAX

status **set_latency_adjustment_options**
[-clocks_to_update *list_of_clocks*]
[-reference_clock *clock_name*]
[-exclude_clocks *list_of_clocks*]
[-ocv_included]

Data Types

list_of_clocks list
clock_name list

ARGUMENTS

-clocks_to_update *list_of_clocks*

Specifies the clocks for which to update the propagated I/O latencies based on the propagated network latency from the clock specified in the **-reference_clock** option. The clocks can be either real clocks or virtual clocks.

-reference_clock *clock_name*

Specifies the clock from which to use the median propagated network latency to calculate the I/O latency for the clocks specified in the **-clocks_to_update** option.

-exclude_clocks *list_of_clocks*

Specifies the clocks to be excluded from the update.

-ocv_included

Specifies that the on-chip variation (OCV) for the updating clocks is already accounted for in their source latency and you do not want to apply extra OCV in the network latency.

By default, the **compute_clock_latency** command copies the network latency of the specified reference clock to the network latency of the updating clocks. If the early and late latencies of the reference clock are different because of OCV, the applied early and late latencies will also be different.

DESCRIPTION

The **set_latency_adjustment_options** command configures the I/O latency updates performed by the **compute_clock_latency** command. You can use this command to exclude clocks from latency updates or to set the reference clock for the updating clocks.

Note that the tool automatically runs the **compute_clock_latency** command as part of the integrated clock tree synthesis and optimization flows. Therefore, it is important to use the **set_latency_adjustment_options** command, even if you do not intend to call the**compute_clock_latency** command explicitly.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following examples show the typical usage.

```
prompt> set_latency_adjustment_options -reference_clock clkA -clocks_to_update {clkX clkY clkZ}
prompt> set_latency_adjustment_options -exclude_clocks  {clkM clkN clkJ}
```

SEE ALSO

clock_opt(2)
compute_clock_latency(2)

set_latency_budget_constraints

set_latency_budget_constraints

Specifies options for treatment of clocks during budgeting.

SYNTAX

int **set_latency_budget_constraints**
[-early_latency *delay*]
[-late_latency *delay*]
[-latency_offset *delay*]
[-source]
[-early_dynamic *delay*]
[-late_dynamic *delay*]
[-clock *budget_clock_spec*]
[-crp *delay*]
[-from_clock *budget_clock_spec*]
[-to_clock *budget_clock_spec*]
[-corner *corner_name* | -default]

Data Types

delay float
budget_clock_spec string
corner_name string

ARGUMENTS

-early_latency *delay*

Specifies the target early clock latency to be used during budgeting. The actual value of this latency is not particularly important. But the relative value of this latency compared to other budgeted clock latencies helps to determine the length of data paths being budgeted. Also, if early latency and late latency are different (implying OCV variance on the clock), this implies that datapath delays will be more tightly constrained. To avoid specifying this value manually, use the **-latency** option of the

compute_budget_constraints command to derive a value. By default, this option applies to all clocks. You can set a different latency in specific blocks and clocks by using the **-clock** option.

-late_latency ***delay***

Specifies the target late clock latency to be used during budgeting. The actual value of this latency is not particularly important. But the relative value of this latency compared to other budgeted clock latencies helps to determine the length of data paths being budgeted. Also, if early latency and late latency are different (implying OCV variance on the clock), this implies that datapath delays will be more tightly constrained.

To avoid specifying this value manually, use the **-latency** option of the **compute_budget_constraints** command to derive a value. By default, this option applies to all clocks. You can set a different latency in specific blocks and clocks by using the **-clock** option.

-latency_offset ***delay***

Specifies that the selected clock should have a different latency from other clocks. For example, setting a latency offset of 1 declares the intent that the clock should be 1ns behind other clocks in the design. Setting a latency offset of -1 declares the intent that the clock should be 1ns ahead of other clocks in the design. Specifying the latency offset is important when you are using the **-balance** option of the **compute_budget_constraints** command. By default, the **-balance** option assumes that all clocks in the design should be exactly balanced to the same latency. Setting latency offset will adjust the balance. You must specify **-clock** with this option.

-clock ***budget_clock_spec***

Applies **-early_latency**, **-late_latency**, or **-latency_offset** to only specific clocks or subtrees of a clock. If the latency is applied to a subtree of a clock, the latency value refers to the latency from the clock source, through the subtree, to the flip-flops. See the "Budget clock specs" section for details on how to select specific clocks.

-crp ***delay***

Sets clock reconvergence pessimism (CRP) in a budgeted timing path. When you have a delay variation (called OCV) in the latency of your clock paths, the early clock latency and late clock latency combine to make your budget paths shorter than they would otherwise be. In some cases, your long and short clock paths share a common set of gates, which causes OCV estimation to be pessimistic. The CRP value is used to offset the pessimism by making your budget paths longer. You must specify **-from_clock** and **-to_clock** with this option.

-source

Use this option along with the **-early_latency** and **-late_latency** options to specify source latency adjustments during the **split_constraints** command. Please refer to the section "Splitting Clock Latencies" below for more details.

-early_dynamic ***delay***

Use this option along with the **-early_latency** and **-source** options. The number specified here will be directly translated to the **-dynamic** option of **set_clock_latency** in the output SDC of the **split_constraints** command. Please refer to the section "Splitting Clock Latencies" below for more details.

-late_dynamic ***delay***

Use this option along with the **-late_latency** and **-source** options. The number specified here will be directly translated to the **-dynamic** option of **set_clock_latency** in the output SDC of the **split_constraints** command. Please refer to the section "Splitting Clock Latencies" below for more details.

-from_clock ***budget_clock_spec***

Applies **-crp** only to paths starting at particular clocks or subtrees of a clock. See the "Budget clock specs" section for details on how to select specific clocks.

-to_clock ***budget_clock_spec***

Applies **-crp** only to paths ending at particular clocks or subtrees of a clock. See the "Budget clock specs" section for details on how to select specific clocks.

-corner ***corner_name***

Specifies that values given for the other options should be applied in the process corner with the specified name.

-default

Specifies that values given for the other options should be applied in process corners do not have specific values given with the **-corner** option.

DESCRIPTION

Specifies options for treatment of clocks during budgeting. These options include the clock latency, OCV variance, CRP, and latency offset. See the option description for more details. By default, budgeting assumes that all of your clocks are aligned and have no OCV. You can change this assumption by using this command. Or you can use **compute_budget_constraints -latency_targets** to generate values automatically. You should set up these parameters before budgeting pins along your datapath, because they have an effect on allowable path delay.

You must supply either **-corner** or **-default** to this command.

Use the **set_budget_options** command to specify general options to be used during creation of timing budgets. To set other specific budget parameters, use the **set_pin_budget_constraints**, and **set_boundary_budget_constraints** commands.

Use the **write_budgets** command to generate SDC that can be applied to lower-level blocks. **compute_budget_constraints** can be used to set key budget parameters automatically. The **report_budget** command generates useful information about the current budget calculation, allocation, and whether that budget is currently being met. Use **write_script -include budget** to save a Tcl script that includes your current settings from this command.

This command returns 1 on success, 0 otherwise.

Budget clock specs

When specifying clocks for various budgeting options, you can provide extra information to limit the parameter to a specific subtree of the clock or edge of a clock. For example, you can set a parameter for the subtree of CLK1 which is inside of BLOCK1. And you can set a different parameter for the subtree of CLK1 which is inside of BLOCK2. In general, every time a clock tree passes into or out of a block, this will define a new part to the tree that you can control.

The syntax for specifying a budget_clock_spec is as follows. To specify all parts of a clock (in all blocks), use the clock name:

CLK1

To specify all clocks in all blocks, use "*":

*****

To specify the part of a clock in a selected block, append ":" and the name of the block instance:

CLK1:my_block

To specify all clocks in a selected block, use "*" followed by ":" and the name of the block instance:

***:my_block**

To specify the part of a clock that crosses into or out of a block at a particular block pin, use the clock name, followed by ":" and the name of the block pin. For example, use the following to specify the top-level portion of CLK1 that originates at the clock

output of my_block:

CLK1:my_block/CLKOUT

To specify a particular edge of a clock, use a second ":" followed by the word "rise" or "fall". This can be used in combination with any of the specifications above that name a particular clock. For example:

CLK1::rise
CLK1:my_block:fall
CLK1:my_block/CLKOUT:rise

Sometimes it is convenient to specify constraints for a master clock and all of the generated clocks downstream from the master clock. We refer to this as a constraint "clock group". To specify a clock group, follow the name of the master clock by a plus (+) character. The specify clock must a be regular clock, not a generated clock. The downstream generated clocks will be implicitly included in the group. Here are the legal usages of clock groups:

CLK1+
CLK1+::rise
CLK1+::fall

If this command is called multiple times for the same clock, the more specific parameter is applied. For example, if you set a parameter for "CLK:my_block" and you set a parameter for "CLK", the parameter for "CLK:my_block" will be used inside my_block and the parameter for "CLK" will be used everywhere else. More general parameters will not overwrite more specific ones.

Splitting Clock Latencies

By default, the **split_constraints** command does not attempt to apportion clock latencies across blocks. Clock latencies are pushed from the chip level down to each block, without change. For example, if the source clock latency of clock CLK1 is 2 at the top level, the source clock latency will still be 2 at the port boundary in the split constraints for each block.

You can use the **plan.budget.split_latencies** app option to cause the **split_constraints** command to reapportion and/or redefine clock latencies at block boundaries. This way you can adjust both the source latency and network latency seen during the synthesis of individual blocks.

Use the **-early_latency**, **-late_latency**, **-source**, **-early_dynamic**, and **-late_dynamic** options of **set_latency_budget_constraints** to control how latencies will be split. latencies should be assigned.

Please refer to the man page of plan.budget.split_latencies for more details.

Multicorner-Multimode Support

EXAMPLES

The following example sets a latency offset for clock CLK in block core/mmu:

```
prompt> set_latency_budget_constraints -latency_offset 0.2 \
    -clock CLK:core/mmu -default
```

The following example sets latency targets for the portion of clock CLK whose source in block core/cpu is "clockin". Note that this pin is the source in this block, not necessarily the global source.

```
prompt> set_latency_budget_constraints -early_latency 3.8 -late_latency 4.2 \
    -clock CLK:core/cpu/clockin -corner C1
```

Same as the last example, but only applies to the rising edge of the clock.

```
prompt> set_latency_budget_constraints -early_latency 3.8 -late_latency 4.2 \
    -clock CLK:core/cpu/clockin:rise -corner C1
```

The following example sets CRP on paths from and to specific clock subtrees in specific blocks.

```
prompt> set_latency_budget_constraints -crp 0.2 -corner C2 \
    -from_clock CLK:core/cpu \
    -to_clock CLK:core/mmu
```

The following example selects values for splitting clock latencies to be used by split_constraints.

```
prompt> set_latency_budget_constraints -clock CLK1 \
        -early_latency 0.50 -late_latency 0.60 -default
prompt> set_latency_budget_constraints -clock CLK1:B1 -source \
        -early_latency 0.42 -late_latency 0.50 -default
prompt> set_app_options -name plan.budget.split_latencies -value prects
prompt> split_constraints
```

SEE ALSO

compute_budget_constraints(2)
report_budget(2)
set_boundary_budget_constraints(2)
set_budget_options(2)
set_budget_shell_latencies(2)
set_pin_budget_constraints(2)
write_budgets(2)
write_script(2)

set_layer_map_file

Saves a layer mapping file into a library.

SYNTAX

status **set_layer_map_file**
-format gds | starrc
[-library *library_name*]
-map_file *layer_map_file*

Data Types

library_name string
layer_map_file string

ARGUMENTS

-format gds | **starrc**

Specifies the format of the layer mapping file, either **gds** for GDSII or **starrc** for StarRC.

-library ***library_name***

Specifies the name of the library in which to store the layer mapping file. If not specified, it uses the current open library. This option is available only an implementation tool. The library manager (lm_shell) always saves the file in the current open library.

-map_file ***layer_map_file***

Specifies the name of the layer mapping file to save into the library.

DESCRIPTION

This command saves a layer mapping file into a library. In that case, the saved file is used as the default layer mapping file when you perform layout validation with the IC Validator tool or parasitic extraction with the StarRC tool.

EXAMPLES

The following example saves a GDS layer mapping file named gds.map into the library named testLib:

```
prompt> set_layer_map_file \
    -format gds \
    -library testLib \
    -map_file gds.map
```

SEE ALSO

remove_layer_map_file(2)
signoff_check_drc(2)
write_gds(2)

set_legalizer_preroute_keepout

This command specifies the keepout margin that is applied to prerouted net shapes during legalization and check_legality. The keepouts may optionally be set to impact only a collection of library pins or instance pins. The keepouts are not persistent in NDM.

SYNTAX

status **set_legalizer_preroute_keepout**
-layer *metal layer name*
-type *prerouted net shape type [strap | via]*
-keepout *keepout {x_keepout y_keepout}*
[-lib_pins] *library pins*
[-pins] *instance pins*

Data Types

layer_name string
type string
keepout pair_of_double
lib_pins collection
pins collection

ARGUMENTS

-layer *layer_name*

Specifies the metal routing layer name, as defined in the technology file

-type *type*

Specifies the type of prerouted net shape the keepout is to be applied. Allowed keywords are "strap" and "via".

-keepout *keepout {x_keepout y_keepout}*

Specifies the keepout values in X and Y directions independently. The keepout values are specified in user units.

-lib_pins *library pins*

Specifies the collection of library pins that are impacted by the set keepout. Optional option. Mutually exclusive with "-pins" option.

-pins *instance pins*

Specifies the collection of instance pins that are impacted by the set keepout. Optional option. Mutually exclusive with "-lib_pins" option.

DESCRIPTION

This command specifies the keepout margin that is applied to prerouted net shapes during all legalization instantiations, including legalize_placement, check_legality, as well as during meta_commands, such as place_opt, clock_opt, etc.

EXAMPLES

The following examples show how to specify keepout on M2 straps (first) and M3 vias (second).

prompt> **set_legalizer_preroute_keepout -layer M2 -type strap -keepout {0.1 0.05}**

prompt> **set_legalizer_preroute_keepout -layer M2 -type via -keepout {0.05 0.08}**

The following examples show how to specify keepout on M2 straps for lib_pins (first) or pins (second).

prompt> **set_legalizer_preroute_keepout -layer M2 -type strap -keepout {0.1 0.05} -lib_pins [get_lib_pins */AND*/Z]**

prompt> **set_legalizer_preroute_keepout -layer M2 -type strap -keepout {0.1 0.05} -lib_pins [get_pins core/*/Z]**

SEE ALSO

legalize_placement(2)
check_legality(2)

set_level_shifter

Sets a strategy for level shifting during implementation.

SYNTAX

status **set_level_shifter**
level_shifter_name
-domain *domain_name*
[-elements *list*]
[-exclude_elements *exclude_objects*]
[-applies_to inputs | outputs | both]
[-source *source_supply_set_name*]
[-applies_to_boundary upper | lower | both]
[-sink *sink_supply_set_name*]
[-threshold *value*]
[-rule low_to_high | high_to_low | both]
[-location self | parent | automatic]
[-input_supply input_supply_set_name]
[-output_supply output_supply_set_name]
[-no_shift] [-force_shift]
[-name_prefix *prefix*]
[-name_suffix *suffix*]
[-update]

Data Types

level_shifter_name string
domain_name string
list list
exclude_objects list
source_supply_set_name string
sink_supply_set_name string
input_supply_set_name string
output_supply_set_name string
value float
prefix string
suffix string

ARGUMENTS

level_shifter_name

Specifies the level shifter strategy name, which is used only for reporting. This argument is required.

-domain *domain_name*

Specifies the domain for which the strategy is applied. This argument is required.

-elements *list*

Specifies a list of design elements, pins, or ports to which this strategy is applied.

-exclude_elements *exclude_objects*

Specifies the objects to which this strategy won't be applied. The objects can be pins of the root cells of the power domain or ports of the top level design for top level power domains.

-applies_to inputs | outputs | both

Specifies whether the given **set_level_shifter** command applies to all inputs or outputs or both types of ports of the power domain. The default value for this option is **both**.

Use following app option to turn off the default applies_to as always both:

>set_app_options -as_user_default -list {mv.upf.ls_default_applies_to_both false}

The default will follow following rule:

1. BOTH : if specified with any filter (-source/-sink/-diff_supply_only), -element, -exclude_elements.

2. OUTPUT : otherwise.

-applies_to_boundary upper | lower | both

Specifies the domain boundary that the given **set_level_shifter** command applies to. The default value for this option is **both** for a domain with the lower domain-boundary design attribute on, and **upper** for domains that do not have the lower domain-boundary design attribute set or have the lower domain-boundary attribute set to off.

-source *source_supply_set_name*

Specifies the source supply set that applies to the elements of the strategy. It filters the ports and pins receiving a net that is driven by the logic that is powered by the supply set.

-sink *sink_supply_set_name*

Specifies the sink supply set that applies to the elements of the strategy. It filters the ports and pins driving a net that fans out to the logic that is powered by the supply set.

-threshold *value*

Specifies the voltage threshold (in volts) for determining when level shifters are required. The default is 0.

-rule low_to_high | high_to_low | both

Specifies which type of level shifters are required. The default is both.

-location self | parent | automatic

Specifies where the level shifter is placed in the logic hierarchy. The default is automatic.

-input_supply *input_supply_set_name*

Specifies the input supply set that applies to the elements of the strategy. Level shifter cell inserted base on this strategy should connect the input supply with the specified supply.

-output_supply *output_supply_set_name*

Specifies the output supply set that applies to the elements of the strategy. Level shifter cell inserted base on this strategy should connect the output supply with the specified supply.

-no_shift

Prevents the insertion of level shifters on the specified ports, pins, and nets when used with the **-elements** option.

The **-no_shift** and **-force_shift** options are mutually exclusive; you must use only one.

-force_shift

Unconditional insertion of a level shifter on the specified elements.

The **-no_shift** and **-force_shift** options are mutually exclusive; you must use only one.

When **-force_shift** is specified with **-threshold**, the **-threshold** is dropped.

-name_prefix *prefix*

Specifies the prefix to use for the level-shifter cell names.

-name_suffix *suffix*

Specifies the suffix to use for the level-shifter cell names.

-update

Indicates that this command adds *elements* or *exclude_elements* for a previous command with the same *strategy_name* and *domain_name* and executed in the same scope.

With the **-update** option, the command only accepts **-elements** or **-excluded_elements**, and the required options of *strategy_name* and **-domain**. If other options are specified with **-update**, the command will error out.

DESCRIPTION

The **set_level_shifter** command is used to set a strategy for level shifting during implementation. Level shifters are placed on the signals that have sources and sinks operating at different voltages, as their associated design elements are connected to different supply nets. If a level shifter strategy is not specified on a power domain, the default level shifter strategy consists of all elements in the power domain, and uses the default strategy settings. Power state table (PST) and operating conditions are considered to determine if level shifters are needed for design elements on the boundaries of power domains. If **-elements** *list* is specified, the elements must be in the domain specified by **-domain** *domain_name*. The **-threshold** option defines how large the voltage difference between the driver and the sink needs to be before level shifters are inserted. Normally, this threshold value is determined from the cell libraries. Use the **-threshold** option to override the library values. The **-rule** value can be low_to_high, high_to_low, or both.

- If low_to_high is specified, signals going from a lower voltage to a higher voltage get a level shifter when the voltage difference exceeds that specified by **-threshold**.
- If high_to_low is specified, signals going from a higher voltage to a lower voltage get a level shifter when the voltage difference exceeds that specified by **-threshold**.
- If both is specified, it is equivalent to having specified both rules in the strategy.

The **-location** option defines where the level shifter cells are placed in the logic hierarchy. All necessary supplies need to be available in the specified location. The option values are as follows:

- If -input/-output_supply is(are) specified, the level shifter cell inserted base on this strategy must have the input/output

power/ground supplies connected with the coresponding supply(ies). If the specified supply(ies) is(are) not available in the candidate domain of the level shifter suppposed to be inserted, the insertion will not happen. User needs to make the supply(ies) to be used available explicitely. If the -input/-output_supply specified will cause new violations, no level shifter will be inserted if the new violation cannot be fixed.

- self specifies that the level shifter cell is placed inside the model or cell being shifted.
- parent specifies that the level shifter cell is placed in the parent of the cell or model being shifted.
- automatic specifies that the implementation tool chooses the appropriate locations.

Level shifter insertion is normally conditional, and level-shifters will not always be inserted even when a strategy is present. With the -force_shift option, level-insertion becomes mandatory. In the following situations, level shifters will be inserted only if the -force_shift option is used:

- Cases where there is no voltage violation. This can mean the driver and loads of the net operate at the same voltage, or the voltage difference between them is less than that specified by the **-threshold** option.
- Cases where there is no real driver or load. If no real driver or load for the net can be found, then the domain primary supply will be used to force level shifter insertion. If no supplies are available, a level shifter will not be inserted.
- Cases where the driver is logic-zero. With **-force_shift**, a level shifter will be inserted with the domain primary as the driver supply.

The **-update** option adds design elements to the -elements and -exclude_elements of an existing level shifter strategy, as if they had been specified with the original command. Adding design elements to the -elements list is allowed only if the first **set_level_shifter** command which defined the strategy also included the **-elements** option. In other words, a domain-based strategy cannot be changed to an elements-based strategy. The level-shifter name, strategy name, domain name, and at least one of the **-elements** or **-exclude_elements** options are required when specifying the **-update** option. No other option is allowed.

Note that this command does not apply to inout ports. Also, it is an error if the specified location is not within the logic design starting at the design root.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the level shifter strategy on PowerDomainZ:

```
prompt> set_level_shifter shift_up -domain PowerDomainZ -applies_to outputs \
    -threshold 0.02 -rule both
```

SEE ALSO

add_port_state(2)
add_pst_state(2)
create_pst(2)
report_pst(2)

set_lib_cell_purpose

Specifies valid purposes for library cells.

SYNTAX

status **set_lib_cell_purpose**
[-include *include_list*]
[-exclude *exclude_list*]
[-reset]
lib_cell_list

Data Types

include_list list
exclude_list list
lib_cell_list collection

ARGUMENTS

-include *include_list*

Specifies the purposes for which the library cells specified in the *lib_cell_list* argument can be used. Valid values are

- **cts** for clock tree synthesis
- **optimization** for delay and electrical design rule optimization
- **hold** for hold fixing
- **power** for power optimization
- **all** for no restrictions on inserting the specified cells
- **none** to prevent insertion of the specified cells for any purpose

This option is mutually exclusive with the **-exclude** option; you must specify one of these options.

-exclude *exclude_list*

Specifies the purposes for which the library cells specified in the *lib_cell_list* argument cannot be used. Valid values are

- **cts** for clock tree synthesis
- **optimization** for delay and electrical design rule optimization
- **hold** for hold fixing

- **power** for power optimization

This option is mutually exclusive with the **-include** option; you must specify one of these options.

-reset

Resets the purposes of the specified library cells. In the library manager (lm_shell), this sets their purposes to all. In an implementation tool, this removes their design-specific purpose overrides from the current block.

This option is mutually exclusive with the **-include** and **-exclude** options.

lib_cell_list

Specifies the affected library cells. The specified library cells must be in a library already loaded into the tool.

Library cell names must contain a library prefix. If more than one object is included, they must be enclosed in quotes or braces ({}).

DESCRIPTION

Defines the valid purposes for the specified library cells.

To remove the purpose information set by this command, use the **set_lib_cell_purpose -include all $lib_cell** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command specifies that the G1 and G2 library cells in the tech_lib library should never be inserted by the tool:

prompt> **set_lib_cell_purpose -include none {tech_lib/G1 tech_lib/G2}**

The following command specifies that the library cells in the tech_lib library whose names start with "buf" should not be used for clock tree synthesis:

prompt> **set_lib_cell_purpose -exclude cts {tech_lib/buf*}**

The following command specifies that the delaybuf library cell in the tech_lib library should be used only for hold fixing:

prompt> **set_lib_cell_purpose -include hold {tech_lib/delaybuf}**

SEE ALSO

get_attribute(2)
report_lib(2)

set_libcell_subset

Sets the library cell subset for sequential cells or instantiated combinational cells to be used for optimization

SYNTAX

status **set_libcell_subset**
-object_list *cells*
-family_name *name*

Data Types

cells list
name string

ARGUMENTS

-object_list *cells*

Specifies a list of sequential cells or instantiated combinational cells for which the specified library cell family is used during optimization. The cells must be either unmapped or instantiated to one of the library cells in the specified library cell family.

-family_name *name*

Specifies the library cell family to be used for optimization of these cells. A family is a subset of library cells and is defined by the **define_libcell_subset** command.

DESCRIPTION

The **set_libcell_subset** command specifies a family of library cells that are available for use during the optimization of the specified instances. A family is defined by the **define_libcell_subset** command.

To remove the library cell family from an instance, use the **remove_libcell_subset** command.

EXAMPLES

The following example sets the library cell subset for the u1 and u2 instances to special_flops, which consists of lib1/SDFLOP1 and lib1/SDFLOP2.

```
prompt> define_libcell_subset \
  -libcell_list [get_lib_cells lib1/SDFLOP1 lib1/SDFLOP2] \
  -family_name special_flops
prompt> set_libcell_subset \
  -object_list [get_cells u1 u2] \
  -family_name special_flops
```

SEE ALSO

remove_libcell_subset(2)
report_libcell_subset(2)
set_libcell_subset(2)
define_libcell_subset(2)

set_load

Sets the capacitance to a specified value on the specified ports in the current design.

SYNTAX

Boolean **set_load**
[-min]
[-max]
[-rise]
[-fall]
[-subtract_pin_load]
[-pin_load]
[-wire_load]
value
objects

value float
objects list

ARGUMENTS

-min

Indicates that the *capacitance* is the minimum capacitance. Applies only to designs in min-max mode (minimum and maximum operating conditions).

-max

Indicates that the *capacitance* is the maximum capacitance. Applies only to designs in min-max mode (minimum and maximum operating conditions).

-rise

Indicates that the *capacitance* is the rise capacitance. This option can be used in combination with **-min** or **-max**. Use this option only with ports. An error message is generated if the *objects* list contains nets.

-fall

Indicates that the *capacitance* is the fall capacitance. This option can be used in combination with **-min** or **-max**. Use this option only with ports. An error message is generated if the *objects* list contains nets.

-subtract_pin_load

This option exists for backwards compatibility, but is ignored.

-pin_load

Indicates that the specified *capacitance* is a pin capacitance. Pin capacitance is not subject to "scaling" using tree_type. Use this option only with ports. An error message is generated if the *objects* list contains nets.

If you do not specify either the **-pin_load** or the **-wire_load** option, or both, **-pin_load** is the default.

-wire_load

Indicates that the specified *capacitance* is a wire capacitance. Pin capacitance is subject to "scaling" using tree_type. Use this option only with ports. An error message is generated if the *objects* list contains nets.

If you do not specify either the **-pin_load** or the **-wire_load** option, or both, **-pin_load** is the default.

value

Specifies the capacitance value to be set on the ports in *objects*. It is in the units of the technology library.

objects

Specifies a list of ports in the current design, on which the *capacitance* is to be set. For backwards compatibility, net objects are accepted, but they are ignored, and a warning message is generated.

DESCRIPTION

This command sets the capacitance to a specified value on specified ports in the current design. If the current design is hierarchical, you must link it using the **link** command. Depending on the options used, **set_load** sets these attributes on the specified objects:

pin_capacitance_max
pin_capacitance_min

If this command is issued without any options, the **pin_capacitance_max** attribute is set; in min-max mode, the **pin_capacitance_min** attribute is also set.

For ports, if **set_load** is issued with only the **-wire_load** option, then the **wire_capacitance_max** attribute is set on the specified ports; in min-max mode, the **wire_capacitance_min** attribute is also set. However, the value is actually counted as part of the total wire capacitance and not as part of the pin/port capacitance.

In min-max mode, using either the **-min** or **-max** option sets only the ***_min** or ***_max** attributes, respectively.

If **-rise** is specifed, the **pin_capacitance_max_rise** and **pin_capacitance_min_rise** attributes are set. You can use this option in conjunction with **-min** or **-max** to set only one of the two attributes. The same conditions apply for the **-fall** option and the **pin_capacitance_*_fall** attributes.

To view capacitance values on ports, use the **report_port** command. To remove a capacitance value, set it to zero. You can remove annotated capacitances from the full design by using the **reset_design** command.

Multicorner-Multimode Support

This command works only on the current corner.

EXAMPLES

The following example sets a capacitance of *2* units on the port named *the_answer*.

prompt> **set_load 2 the_answer**

The following example sets a wire capacitance of *5* units on the port named *the_answer*.

prompt> **set_load -wire_load 5 the_answer**

SEE ALSO

all_outputs(2)
current_design(2)
report_port(2)
reset_design(2)
set_drive(2)

set_locked_objects

Sets the specified objects to the locked state. The command sets the value of **physical_status** attribute to either **locked** or **unrestricted**;

SYNTAX

status **set_locked_objects**
[-unlock]
[*object_list*]

Data Types

object_list collection

ARGUMENTS

-unlock

Unlocks the specified objects.

object_list

Specifies the collection or selection set of objects for which to change locked status. If this option is not specified, the global selection set is used.

DESCRIPTION

This command sets the locked status for the specified objects. If no objects are specified, the global selection set is used.

EXAMPLES

The following example locks all selected objects.

```
prompt> set_locked_objects
1
```

The alternative syntax uses collection to specify objects.

```
prompt> set_locked_objects [get_selection]
1
```

The following example locks all hard macros.

```
prompt> set_locked_objects [get_cells -physical_context -filter "is_hard_macro && !is_physical_only"]
1
```

SEE ALSO

set_fixed_objects(2)
change_selection(2)
get_selection(2)

set_logicbist_configuration

set_logicbist_configuration

Specifies the LogicBIST compression configuration for the design.

SYNTAX

status **set_logicbist_configuration**
[-chain_count *chain_count*
| -max_length *max_chain_length*]
[-clock *clock_name*]
[-xtolerance high | extended | low]
[-pattern_counter_width *register_width*]
[-shift_counter_width *register_width*]
[-prpg_width *register_width*]
[-misr_width *register_width*]
[-occ_clock_weights *weight_list*]
[-test_mode *logicbist_mode_name*]

Data Types

chain_count integer
max_chain_length integer
clock_name string
register_width integer
weight_list list
base_mode_name string
logicbist_mode_name string

ARGUMENTS

-chain_count *chain_count*

Specifies the number of scan chains to build. The tool attempts to meet this constraint. If unable to do so, the tool issues a warning. See the DESCRIPTION section for more information.

-max_length ***max_chain_length***

Specifies the maximum allowed length of the compressed scan chains. The tool attempts to meet this constraint. If unable to do so, the tool issues a warning. See the DESCRIPTION section for more information.

-xtolerance high | **extended** | **low**

Specifies whether to enable XLBIST or Extended Xtol capability. This option is mandatory, and **low** would be supported in future.

-clock ***clock_name***

Specifies the scan clock to be used for the LogicBIST codec (compressor and decompressor) to be inserted. The specified clock must be previously declared as a test clock using the **set_dft_signal** command. By default, the tool selects the first-defined scan clock.

-pattern_counter_width ***register_width***

Specifies the width of the pattern counter register. The maximum number of patterns is determined by the width of this register as follows:

max_patterns == (2 ^ pattern_counter_register_width) - 2

The default is 8, which allows a maximum of 254 patterns.

-shift_counter_width ***register_width***

Specifies the width of the shift counter register, which counts down the shift cycles for each pattern.

By default, the tool sizes this register according to the longest shift chain in the design. You normally do not need to specify this value.

-prpg_width ***register_width***

Specifies the width of the pseudo-random pattern generator (PRPG) register. This register generates pattern data that feeds the compressed scan chains through an XOR phase shifter. The PRPG seed value is also this width.

By default, the tool sizes this register using a heuristic based on the number of compressed scan chains.

-misr_width ***register_width***

Specifies the width of the multiple-input signature register (MISR). This register captures data from the compressed scan chains through an XOR compactor. The final MISR signature value is also this width.

If this register width is set to the same value as the number of compressed scan chains, the tool does not implement an XOR compactor.

By default, the tool sizes register this using a heuristic based on the number of compressed scan chains.

-occ_clock_weights ***weight_list***

Specifies a list of clock signal groups and their weights.

By default, all OCC clocks capture in each pattern. If asynchronous cross-clock paths exist, you can use this option to group clocks into capturing groups. Each capture group has a weight value that specifies its probability of capture relative to the other groups.

Each group is itself a list containing an integer weight value followed by one or more clock signals:

-occ_clock_weights {{75 SYSCLK} {25 CLK33 CLK66}}

-test_mode ***logicbist_mode_name***

Specifies the LogicBIST test mode to which the specification applies. The default is *all_dft*, except when **define_test_mode** is

used. In this case, the default is the value of the last *test_mode* defined.

Note that this is the test mode used by TetraMAX for seed/signature computation; it is not the test mode used for autonomous self-test operation.

DESCRIPTION

The **set_logicbist_configuration** command allows you to specify certain LogicBIST insertion parameters prior to using the **insert_dft** command. The parameters include specifications such as compressed scan chain count, pattern count register width, and MISR and PRPG register widths.

There are two related options to control the number of compressed scan chains for LogicBIST insertion, **-chain_count** and **-max_length**. You must specify one of these options to configure LogicBIST compression. If both are specified, the **-max_length** option takes precedence over the **-chain_count** option.

SEE ALSO

define_test_mode(2)
insert_dft(2)
preview_dft(2)

set_macro_constraints

Sets constraints on hard macro or I/Os cells.

SYNTAX

status **set_macro_constraints**
[-allowed_orientations *orientation_list*]
[-preferred_location *relative_location*]
[-alignment_grid *grid_name*]
[-alignment_point *align_point*]
[-align_pins_to_tracks]
[-alignment_orientation_set all | R0 | R90]
hard_macro_or_IO_list

Data Types

orientation_list	list of orientations in DEF syntax
relative_location	pair of relative ordinates in interval [0, 1]
grid_name	grid name
align_point	point in the macro master coordinates
hard_macro_or_IO_list	list of hard macros or I/Os

ARGUMENTS

-allowed_orientations *orientation_list*

Specifies one or more allowed placement orientations for the specified list of cells. The list of orientation values can contain one or more of the following values: R0, R90, R180, R270, MX, MXR90, MY, MYR90. The behavior is different between hard macros and I/Os. For hard macros, the list of legal orientations defined in the library takes precedence over this option, and the list of orientations specified here further restricts the legal placement orientations. For I/Os, the setting made here overrides the legal orientations setting in the library. This option is optional.

-preferred_location *location_list*

Specifies the preferred location that is applied to the given list of hard macros. The location is specified in the coordinate system relative to the encompassing block, where coordinate (0, 0) is mapped to the lower-left corner of the block's bounding box, while (1, 1) is mapped to the upper right corner of the block's bounding box. This option is optional,

-alignment_grid *grid_name*

Specifies the name of the grid to align macros to. This option is optional and is mutually exclusive with the **-align_pins_to_tracks** option. If **-align_pins_to_tracks** has been previously specified for the same macro and orientation, specifying an alignment grid removes the **-align_pins_to_tracks** setting.

-alignment_point *align_point*

Specifies the point, in the macro master coordinates, that must be on grid. The default is the origin. This option requires the **-alignment_grid** option.

-align_pins_to_tracks

If this option is specified, the command aligns signal pins to the grid derived based on the tracks on the layers that the pin shape belongs to. If there is track color (mask) and pin color information, the color-matching rules will be enforced. This option is optional and is mutually exclusive with the **-alignment_grid** and **-alignment_point** options. If an alignment grid has been specified previously for the same macro and orientation, specifying this option removes the grid specification.

-alignment_orientation_set all | **R0** | **R90**

Specifies the set of macro orientations with the same footprint that the grid applies to, either R0 = {R0, R180, MX, MY} or R90 = {R90, R270, MXR90, MYR90}. Each set is denoted by its canonical representative, R0 or R90. This option is optional. The default is all. Requires one of **-alignment_grid** and **-align_pins_to_tracks**.

hard_macro_list

Specifies the list of hard macros for which to create constraints. This list of hard macros is required.

DESCRIPTION

This command is used to set placement constraints on hard macros or I/Os. These constraints are respected by the **create_placement -floorplan** command.

EXAMPLES

The following example sets two possible orientations for the specified cell.

```
prompt> set_macro_constraints -allowed_orientations {R0 R180} [get_selection]
1
```

The following example sets the preferred location for the selected macro to the center bottom of the encompassing block it belongs to.

```
prompt> set_macro_constraints -preferred_location {0.5 0} [get_selection]
1
```

SEE ALSO

create_placement(2)
derive_preferred_macro_locations(2)
report_macro_constraints(2)
remove_macro_constraints(2)

set_macro_relative_location

Specifies a constraint to place a hard macro or macro array relative to an anchor object.

SYNTAX

status **set_macro_relative_location**
-target_object *target_object*
-target_orientation *target_orientation*
-target_corner bl | br | tl | tr
[-anchor_object *anchor_objecf*]
-anchor_corner bl | br | tl | tr
-offset *xy_distance* | *xy_factor* | {*x_distance*
y_distance} | {*x_factor* | *y_factor*}
[-offset_type { fixed|scalable fixed|scalable }]
[-used_length *xy_distance* | {*x_distance y_distance*}]
[-scale_edge *xy_edge* | {*x_edge y_edge*}]

Data Types

target_object hard macro, macro array edit group
target_orientation orientation in DEF syntax
anchor_object IO pad cell, hard macro, block, port, pin, macro
array edit group, move bound, voltage area, voltage area shape, core area
x_distance float
y_distance float
xy_distance float
x_factor float
y_factor float
xy_factor float
x_edge unsigned integer
y_edge unsigned integer
xy_edge unsigned integer

ARGUMENTS

-target_object *target_object*

Specifies the required hard macro or macro array edit group for which to create macro relative location constraint. The target object should not be fixed.

-target_orientation *target_orientation*

Specifies the allowed placement orientation for the hard macro cell. The orientation value can contain one of the following values: R0, R90, R180, R270, MX, MXR90, MY, MYR90. The orientation specified here further restricts the legal placement

orientation, and should not be conflict with the list of legal orientations defined in the library or set by command **set_macro_options**. For macro array, the target orientation is always R0.

-target_corner bl | tl | tr | br

Specifies which corner of the target macro or macro array to apply the constraint. Valid target corner values are: bl, tl, tr, br. These four keywords represent bottom-left, top-left, top-right, bottom-right respectively.

-anchor_object *anchor_object*

Specifies the anchor object for which to create macro relative location constraint on target hard macro or macro array. The anchor object can be IO pad cell, hard macro, block, port, pin, macro array edit group, move bound, voltage area or voltage area shape, core area and should be in the same design with target hard macro or macro array. By default the anchor object is the target object's current parent block.

-anchor_corner bl | tl | tr | br | 1 | 2 | 3 | 4 | 5 | ...

Specifies which corner of the boundary bbox for rectangle anchor object or which corner for rectilinear anchor block to apply the constraint. Valid anchor corner values are: bl, tl, tr, br for rectangle or 1, 2, 3, 4, 5... for each corner of a shape. A rectilinear corner number is a positive integer that starts at one, and increments by one as you proceed around each edge in the shape clockwise. Given any rectilinear shape, the bottom corner of the lowest left-most edge is the starting corner (corner number 1). You cannot specify a value of 0 for this option. The keywords bl, tl, tr, br represent bottom-left, top-left, top-right, bottom-right respectively. The boundary bbox is used for macro and block, and bounding box for other anchor objects.

-offset *xy_distance* | *xy_factor* | {*x_distance*

y_distance} | {*x_factor* | *y_factor*}" Specifies the distance from the target corner to the anchor corner in X and Y dimension or a factor controlled by -offset_type option. The distance value can be positive, negative or zero. A positive value would indicate the target corner being on the right/top side of the anchor corner, and a negative value would result in the target corner being on the left/bottom side of the anchor corner, while a value of zero would mean the target corner has the same coordinate as the anchor corner. If option -offset_type is set "scalable" then this option value is a factor, the actual_offset = factor * (scale_edge_length - used_length_as_specified _in_set_macro_relative_location). The calculation may have some precision loss.

-offset_type { fixed|scalable fixed|scalable }

Specifies whether the value set in option -offset is the distance from the target corner to the anchor corner in X and Y dimension or a factor. Default is fixed type.

-used_length *xy_distance* | {*x_distance y_distance*}

Specifies the macro occupied distance in X and Y dimension according to the scale_edge of block/core_area/movebound/voltage_area_shape.

-scale_edge *xy_edge* | {*x_edge y_edge*}

Specifies scale edge number in X and Y dimension. The scale edge number should be 1, 2, 3... for each edge. A edge number is a positive integer that starts at 1, and increments by one as you proceed around each edge in the shape clockwise. Given any rectilinear shape, the lower leftmost vertical edge is the starting edge (edge number 1). Scale_edge is for the scalable offset. When offset_type is scalable, the actual offset value in terms of microns will be calculated at the time of macro placement, using the length of the parent_block/core_area/move_bound(if anchor object is movebound)/voltage_area_shape(if anchor object is VA) edge specified through scale_edge option and the value from offset option. If user doesn't specify scale_edge with offset scalable, the width/height of the block/core_area/ movebound/voltage_area_shape boundary bbox is used instead of the length of any particular edge.

DESCRIPTION

This command is used to set relative location constraint on hard macro or macro array relative to an anchor object. The location of the macro or macro array is determined by the parameters given. These placement constraints are respected by the **create_placement -floorplan** command.

EXAMPLES

The following example specifies a relative location constraint on the top-right corner of hard macro T1 with respect to the top-left corner of anchor macro A1.

```
prompt> set_macro_relative_location -target_object [get_cell T1]
        -target_orientation R180 -target_corner tr
        -anchor_object [get_cell A1] -anchor_corner tl
        -offset {150 100}
1
```

The following example specifies a relative location constraint on the bottom-right corner of hard macro T1 with respect to the bottom-left corner of anchor port P1.

```
prompt> set_macro_relative_location -target_object [get_cell T1]
        -target_orientation R270 -target_corner br
        -anchor_object [get_port P1] -anchor_corner bl
        -offset {300}
1
```

The following example specifies a relative location constraint on the top-right corner of macro array T1 with respect to the top-left corner of anchor voltage area P1.

```
prompt> set_macro_relative_location -target_object [create_macro_array
        -num_rows 2 -num_cols 2 [get_selection] -name T1]
        -target_orientation R0 -target_corner tr
        -anchor_object [get_voltage_area P1] -anchor_corner tl
        -offset {0.8 0.9} -offset_type {scalable scalable}
        -used_length {2.355 3.255} -scale_edge {2 1}
1
```

SEE ALSO

report_macro_relative_location(2)
remove_macro_relative_location(2)
derive_macro_relative_location(2)
write_macro_relative_location(2)
create_placement(2)
create_macro_relative_location_placement(2)

set_max_capacitance

Sets the maximum capacitance for ports, library cell pins, leaf cell pins, clocks or designs.

SYNTAX

string **set_max_capacitance**
value
[-clock_path]
[-data_path]
[-modes *mode_list*]
[-corners *corner_list*]
[-db *file_name*]
[-scenarios *scenario_list*]
object_list

Data Types

value	float
mode_list	list
corner_list	list
file_name	string
scenario_list	list
object_list	list

ARGUMENTS

value

Specifies the capacitance limit. The *value* setting must be greater than zero. This is the maximum total capacitance (pin plus wire capacitance) in library capacitance units.

-clock_path

Specifies all pins that are in the network of the specific clocks in *object_list*.

-data_path

Specifies all pins that are in the data paths launched by the specific clocks in *object_list*.

-modes *mode_list*

Specifies the scenarios to which to apply the capacitance limit. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option must not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners *corner_list*

Specifies the scenarios to which to apply the capacitance limit. If this option is given, all scenarios of the given corners and the current mode will be used. The **-scenarios** option must not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-db ***file_name***

Specifies the full or partial DB file name that matches a unique DB file for library cell pins, to which capacitance limit is applied. This option is valid only for library cell pin objects.

-scenarios ***scenario_list***

Specifies the scenarios to which to apply the capacitance limit. The **-modes** or **-corners** option must not be specified together with this option.

object_list

Provides a list of ports, library cell pins, leaf cell instance pins, clocks or designs on which to set maximum capacitance.

DESCRIPTION

Specifies a maximum capacitance on ports, library cell pins, leaf cell instance pins, clocks or designs for the current scenario. If maximum capacitance is set on a port, the net connected to that port is expected to have a total capacitance less than the specified *capacitance_value*. If specified on a design, the default maximum capacitance for that design is set. Library cell pins also can have a maximum capacitance value specified in the library.

If maximum capacitance is specified for library cell pin objects using this command, it overrides the values specified in the library for the specified scenarios. The library cell pin limits are stored as design specific overrides in the scenarios of the current design and are only applicable to the current design.

A maximum capacitance for leaf cell instance pin object creates an implicit size_only attribute on the pin. This is to preserve the constraint and may restrict optimization. It is recommended to minimize number of pin constraints to avoid restricting optimization. Also, pin constraints may not be able to satisfy user intent. A dominating pin constraint on a driver pin will remove driver sizing as possible solution.

Constraining max_capacitance, used in costing, is the most restrictive of design, clock, lib_pin and pin constraint for a pin.

Use the **-db** option to specify library cell pin constraints for a specific DB library that was used to create the NDM library. The command doesn't access the file itself but uses the name to find matching panes to apply the constraints. The **report_lib** and **report_cells** commands can be used to find the DB file names. *file_name* can be a library name, a local DB file name or a full path name that matches a unique DB file. Constraints with **-db** files have higher precedence than without it. Use **remove_max_capacitance** to remove all library cell pin constraints including **-db** files.

If maximum capacitance is set on a clock, the maximum capacitance is applied to all pins in this specified clock domain. Within a clock domain, you can optionally restrict the constraint further to clock paths only or data paths only.

The most restrictive of the design, clock and pin or port limit is used.

Maximum capacitance limits are managed on a per-scenario basis. For designs with multiple scenarios, you can specify different limits for different scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the given limit is applied to the current scenario. The tool issues an error if there is no current scenario. If the **-scenarios** option is given, the limit is applied to all of the specified scenarios.

If the **-modes** option is given, the limit will be applied to all scenarios of the specified modes. The tool issues an error if none of the specified modes have any scenarios.

If the **-corners** option is given, the limit will be applied to all scenarios of the specified corners and the current mode. The tool issues

an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are given, the limit will be applied to all scenarios of the specified corners and the specified modes. The tool issues an error if there are no such scenarios. It is an error to give **-scenarios** with **-modes** or **-corners**.

The **report_ports -design_rule** command shows port maximum capacitance limits. The **report_design** command shows the default maximum capacitance setting for the scenarios of the current design.

To remove maximum capacitance limits from designs or ports, use the **remove_max_capacitance** command.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example sets a maximum capacitance limit of 2.0 units on ports "OUT*".

```
prompt> set_max_capacitance 2.0 [get_ports "OUT*"]
```

The following example sets the default maximum capacitance limit of 5.0 units on the current scenario of the current design.

```
prompt> set_max_capacitance 5.0 [current_design]
```

The following example sets a maximum capacitance limit of 0.8 units on all pins of the clock network of CLK.

```
prompt> set_max_capacitance 0.8 [get_clocks CLK] -clock_path
```

The following example sets a maximum capacitance limit of 0.7 units on all pins of the data path of CLK.

```
prompt> set_max_capacitance 0.7 [get_clocks CLK] -data_path
```

The following example sets a maximum capacitance limit of 0.01 units on lib_pin for the current mode for specified full path name of db files. Full path name is used to uniquely identify slow.db file.

```
prompt> set_max_capacitance 0.01 -db /top/a/b/slow.db [get_lib_pin slow/BUF/Y]
prompt> set_max_capacitance 0.01 -db /top/a/c/slow.db [get_lib_pin slow/BUF/Y]
```

SEE ALSO

current_design(2)
get_ports(2)
remove_max_capacitance(2)
report_design(2)
report_ports(2)
set_max_transition(2)
set_min_capacitance(2)

set_max_delay

Specifies a maximum delay for timing paths.

SYNTAX

status **set_max_delay**
[-rise]
[-fall]
[-reset_path]
[-ignore_clock_latency]
[-from *from_list*
| -rise_from *rise_from_list*
| -fall_from *fall_from_list*]
[-through *through_list*]*
[-rise_through *rise_through_list*]*
[-fall_through *fall_through_list*]*
[-to *to_list*
| -rise_to *rise_to_list*
| -fall_to *fall_to_list*]
[-comment *comment*]
delay_value

Data Types

from_list list
rise_from_list list
fall_from_list list
through_list list
rise_through_list list
fall_through_list list
to_list list
rise_to_list list
fall_to_list list
comment string
delay_value float

ARGUMENTS

-rise

Specifies that the tool constrains only rising path delays. If neither **-rise** nor **-fall** is specified, both rising and falling delays are constrained.

-fall

Specifies that the tool constrains only falling path delays. If neither **-rise** nor **-fall** is specified, both rising and falling delays are constrained.

-reset_path

Removes existing point-to-point exception information from the specified paths. If used with **-to** only, all paths leading to the specified endpoints are reset. If used with **-from** only, all paths leading from the specified startpoints are reset. If used with **-from** and **-to**, only paths between those points are reset. Only information of the same rise or fall setup or hold type is reset. Using this option is equivalent to using the **reset_paths** command with similar arguments before issuing **set_max_delay**.

-ignore_clock_latency

Indicates that the launch and capture clock latencies is to be ignored when computing slack on the specified paths. Note that these paths will be considered as clock-less, and therefore, reporting these paths by their respective ignored clocks will result in unconstrained paths. To report these paths with ignore_clock_latency, the user may specify the launched and/or captured devices as options in the report_timing command.

-from *from_list*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If a cell is specified, one path startpoint on that cell is affected. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the specified objects. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the specified objects. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

Specifies a list of pins, ports, cells, and nets through which the paths must pass for maximum delay definition. Nets are interpreted to imply the leaf-level driver pins. If you omit **-through**, all timing paths specified using the **-from** and **-to** options are affected. You can specify **-through** more than one time in a single command invocation. For a discussion of multiple **-through**, **-rise_through**, and **-fall_through** options, see the DESCRIPTION section.

-rise_through *rise_through_list*

This option is similar to the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than one time in a single command invocation. For a discussion of multiple **-through**, **-rise_through**, and **-fall_through** options, see the DESCRIPTION section.

-fall_through *fall_through_list*

This option is similar to the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than one time in a single command invocation. For a discussion of multiple **-through**, **rise_through**, and **fall_through** options, see the DESCRIPTION section.

-to *to_list*

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-rise_to ***rise_to_list***

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to ***fall_to_list***

Same as the **-to** option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-comment ***comment***

Associates a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

delay_value

Specifies a floating point number that represents the required maximum delay value for specified paths. *delay_value* must have the same units as the technology library used during analysis. If a path startpoint is on a sequential device, clock skew is included in the computed delay. If a path startpoint has an input delay specified, that delay value is added to the path delay. If a path endpoint is on a sequential device, clock skew and library setup time are included in the computed delay. If the endpoint has an output delay specified, that delay is added into the path delay.

DESCRIPTION

This command specifies a required maximum delay for timing paths in the current design. The path length for any startpoint in *from_list* to any endpoint in *to_list* must be less than *delay_value*.

The value of a **max_rise_delay** attribute cannot be less than that of a **min_rise_delay** attribute on the same path (and similarly for fall attributes). If this condition occurs, the older attribute is removed.

Individual maximum delay targets are automatically derived from clock waveforms and port input or output delays. For more information, refer to the **create_clock**, **set_input_delay**, and **set_output_delay** command man pages.

The **set_max_delay** command is a point-to-point timing exception command. For example, the command overrides the default single-cycle timing relationship for one or more timing paths. Other point-to-point timing exception commands include **set_multicycle_path**, **set_min_delay**, and **set_false_path**. A **set_max_delay** or **set_min_delay** command overrides a **set_multicycle_path** command.

The more general commands apply to more than one path. For example, either **-from** or **-to** is used (but not both), or clocks are used in the specification. Within a given point-to-point exception command, the more specific command overrides the more general. The following list of commands is arranged in order, from the highest to the lowest precedence (more specific to more general).

1. set_max_delay -from *pin* -to *pin*
2. set_max_delay -from *clock* -to *pin*
3. set_max_delay -from *pin* -to *clock*
4. set_max_delay -from *pin*
5. set_max_delay -to *pin*
6. set_max_delay -from *clock* -to *clock*

7. set_max_delay -from *clock*

8. set_max_delay -to *clock*

You can use multiple **-through**, **-rise_through**, and **fall_through** options in a single command to specify paths that traverse multiple points in the design. The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

```
-from A1 -through B1 -through C1 -to D1
```

If more than one object is specified within one **-through**, **-rise_through**, or **-fall_through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
-from A1 -through {B1 B2} -through {C1 C2} -to D1
```

To list the **max_delay**, **min_delay**, **multicycle_path**, and **false_path** information for the design, use the **report_exceptions** command.

To remove information set by **set_max_delay**, use the **reset_path** or **reset_design** command.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example specifies that any delay path to port "Y" must be less than 10.0 units.

```
prompt> set_max_delay 10.0 -to {Y}
```

The following example specifies that all paths from ff1a or ff1b to ff2e must have delays less than 15.0 units.

```
prompt> set_max_delay 15.0 -from {ff1a ff1b} -to {ff2e}
```

The following example specifies that all paths to endpoints clocked by PHI2 must have delays less than 8.5 units.

```
prompt> set_max_delay 8.5 -to [get_clocks PHI2]
```

The following example sets a requirement that all paths leading to ports named "busA[*]" must have delays less than 5.0.

```
prompt> set_max_delay 5.0 -to "busA[*]"
```

The following example specifies that all timing paths from ff1/CP to ff2/D that pass through one or more of {U1/Z U2/Z} and one or more of {U3/Z U4/C} must have delays less than 8.0 units.

```
prompt> set_max_delay 8.0 -from ff1/CP -through {U1/Z U2/Z} -through {U3/Z U4/C} -to ff2/D
```

The following example specifies that all timing paths from ff1/CP to ff2/D that rise through one or more of {U1/Z U2/Z} and fall through one or more of {U3/Z U4/C} must have delays less than 8.0 units.

```
prompt> set_max_delay 8.0 -from ff1/CP -rise_through {U1/Z U2/Z} \
  -fall_through {U3/Z U4/C} -to ff2/D
```

SEE ALSO

create_clock(2)
current_design(2)
group_path(2)
reset_design(2)
reset_paths(2)
set_false_path(2)
set_input_delay(2)
set_min_delay(2)
set_multicycle_path(2)
set_output_delay(2)

set_max_fanout

NOTE

set_max_fanout

This command is not supported.

set_max_lvth_percentage

This command specifies a soft constraint for the target percentage of low-threshold-voltage cells. The set_max_lvth_percentage command is deprecated. Use the set_multi_vth_constraint command instead.

SYNTAX

status **set_max_lvth_percentage**
percent_value

Data Types

percent_value float

ARGUMENTS

percent_value

Specifies the target percentage of low-threshold-voltage cells.

DESCRIPTION

This command specifies a target percentage (cell count) for low threshold voltage cells to be used in the design. This is a soft constraint and optimization tries to honor it if timing convergence is not affected by this limit.

Note that the percentage is computed based on cell count.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example specifies that no more than three percent of the design's cell count can be from low-threshold-voltage cells.

```
prompt> set_max_lvth_percentage 3
1
```

SEE ALSO

set_threshold_voltage_group_type(2)
remove_max_lvth_percentages(2)
set_multi_vth_constraint(2)

set_max_time_borrow

Limits time borrowing for latches.

SYNTAX

string **set_max_time_borrow**
delay_value
object_list
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]

float *delay_value*
list *object_list*
list *mode_list*
list *corner_list*
list *scenario_list*

ARGUMENTS

delay_value

Specifies the value to which the **max_time_borrow** attribute is set. Defines the desired limit of time borrowing on the latches specified in **object_list**. *delay_value* must be between zero and the default maximum derived from the waveform. By default, the maximum is derived from the ideal clock waveform driving each latch, and is equal to (closing_edge - open_edge). Library setup and data-to-Q propagation times are automatically taken into account. *delay_value* is in the same units as those in the technology library used during analysis.

object_list

Specifies a list of objects for which time borrowing is to be limited to *value*. The objects can be clocks, latch cells, data pins, or clock (enable) pins. If a cell is specified, all enable pins on that cell are affected.

-modes *mode_list*

Specifies the scenarios that the time borrow limit will be applied to. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option may not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by -corners will be used.

-corners *corner_list*

Specifies the scenarios that the time borrow limit will be applied to. If this option is given, all scenarios of the given corners and the current mode will be used. The **-scenarios** option may not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by -corners will be used.

-scenarios *scenario_list*

Specifies the scenarios that the time borrow limit will be applied to. The **-modes** or **-corners** option may not be given with this option.

DESCRIPTION

Sets the **max_time_borrow** attribute to **value** to constrain the amount of time borrowing possible for level-sensitive latches. To meet delay targets, **set_max_time_borrow** prevents automatic use of all or part of the enabling clock pulse on a latch.

Max_time_borrow settings are managed on a per-scenario basis. For designs with multiple scenarios, you can specify different values for different scenarios by using **-modes**, **-corners**, and **-scenarios** options. By default, if none of these options are specified, the given value is applied to the current scenario. (It will be an error if there is no current scenario). If the **-scenarios** option is given, the value will be applied to all of the specified scenarios. If the **-modes** option is given, the value will be applied to all scenarios of the specified modes. (It will be an error if none of the specified modes have any scenarios.) If the **-corners** option is given, the value will be applied to all scenarios of the specified corners and the current mode. (It will be an error if none of the specified corners have any scenarios with the current mode.) If both the **-modes** and **-corners** options are given, the value will be applied to all scenarios of the specified corners and the specified modes. (It will be an error there are no such scenarios.) It is an error to give -scenarios with **-modes** or **-corners**.

To get **max_time_borrow** attributes on the design, use **get_attribute**.

To undo **set_max_time_borrow**, use **remove_max_time_borrow**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example restricts time borrowing on latches **latch1a** and **latch2f** to 4.0 units.

prompt> **set_max_time_borrow 4.0 { latch1a latch2f }**

The following example specifies that no time borrowing will take place on **latch1c**.

prompt> **set_max_time_borrow 0.0 { latch1c }**

SEE ALSO

remove_max_time_borrow(2)
get_attribute(2)

set_max_transition

Sets maximum transition for ports, library cell pins, leaf cell pins, clocks or designs with respect to the main library trip-points.

SYNTAX

string **set_max_transition**
value
[-clock_path]
[-data_path]
[-modes *mode_list*]
[-corners *corner_list*]
[-db *file_name*]
[-scenarios *scenario_list*]
object_list

Data Types

value float
mode_list list
corner_list list
file_name string
scenario_list list
object_list list

ARGUMENTS

value

Specifies the transition limit. The *value* must by greater than zero. This setting is the maximum transition time in user-defined input time units.

-clock_path

Specifies all pins that are in the network of the specific clocks in the *object_list*.

-data_path

Specifies all pins that are in the data paths launched by the specific clocks in *object_list*.

-modes *mode_list*

Specifies the scenarios to which to apply the transition limit. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option must not be specified together with the **-modes** option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners *corner_list*

Specifies the scenarios to which to apply the transition limit. If this option is given, all scenarios of the specified corners and the current mode will be used. The **-scenarios** option must not be specified together with the **-corners** option. If both **-modes** and **-corners** are specified, the command uses scenarios whose mode is specified by **-modes** and whose corner is specified.

-db ***file_name***

Specifies the full or partial DB file name that matches a unique DB file for library cell pins, to which transition limit is applied. This option is valid only for library cell pin objects.

-scenarios ***scenario_list***

Specifies the scenarios to which to apply the transition limit. The **-modes** or **-corners** option must not be specified together with the **-scenarios** option.

object_list

Provides a list of ports, library cell pins, leaf cell instance pins, clocks or designs on which to set maximum transition.

DESCRIPTION

Specifies a maximum transition on ports, library cell pins, leaf cell pins, clocks or designs for the current scenario. If maximum transition is set on a port, the port must have a transition time less than the specified *transition_value*. If specified on a design, the default maximum transition for that design is set. Library cell pins can also have a **max_transition** value specified in the library.

If you specify a maximum transition for library cell pin objects, the command overrides the values specified in the library for the specified scenarios. The library cell pin limits are stored as design specific overrides in the scenarios of the current design and are only applicable to only the current design.

A maximum transition for leaf cell instance pin object creates an implicit size_only attribute on the pin. This is to preserve the constraint and may restrict optimization. It is recommended to minimize number of pin constraints to avoid restricting optimization. Also, pin constraints may not always be able to satisfy user intent.

Constraining max_transition, used in costing, is the most restrictive of design, clock, lib_pin and pin constraint for a pin.

The **-db** option specifies library cell pin constraints for a specific DB library that was used to create the NDM library. The command does not access the file itself, but uses the name to find matching panes to apply the constraints. The **report_lib** and **report_cells** commands can be used to find the DB file names. The *file_name* argument can be a library name, a local DB file name or a full path name that matches a unique DB file. Constraints with **-db** files have higher precedence than without it. Use **remove_max_transition** command to remove all library cell pin constraints including **-db** files.

The most restrictive of the design, clock and pin or port limit is used. If the user-specified limit is the most restrictive limit, it is scaled to that of the pin trip-points during slack computation.

If maximum transition is set on a clock, the maximum transition is applied to all pins in this specified clock domain. Within a clock domain, you can optionally restrict the constraint further to clock paths only or data paths only, and to rising transitions only or falling transitions only.

The **-clock_path** and **-data_path** options can be used only if the list of objects is a clock list, not a port list or design list. If a clock list is specified without **-clock_path** or **-data_path**, both clock path and data path for both rising and falling transitions are considered by default.

Maximum transition limits are managed on a per-scenario basis. For designs with multiple scenarios, you can specify different limits for different scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the given limit is applied to the current scenario. The tool issues an error if there is no current scenario. If the **-scenarios** option is given, the limit is applied to all of the specified scenarios.

If the **-modes** option is given, the limit will be applied to all scenarios of the specified modes. The tool issues an error if none of the specified modes have any scenarios.

If the **-corners** option is given, the limit will be applied to all scenarios of the specified corners and the current mode. The tool issues an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are given, the limit will be applied to all scenarios of the specified corners and the specified modes. The tool issues an error if there are no such scenarios. It is an error to give **-scenarios** with **-modes** or **-corners**.

The **report_ports -design_rule** command shows the maximum transition limits for the specified ports. The **report_design** command shows the default maximum transition setting for the scenarios of the current design.

To remove maximum transition limits from designs or ports, use **remove_max_transition**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example sets a maximum transition limit of 2.0 units on ports "OUT*".

```
prompt> set_max_transition 2.0 [get_ports "OUT*"]
```

The following example sets the default maximum transition limit of 5.0 units on the current scenario.

```
prompt> set_max_transition 5.0 [current_design]
```

The following example sets a maximum transition limit of 2.0 units on all pins in the clock network of CLK.

```
prompt> set_max_transition 2.0 [get_clocks CLK] -clock_path
```

The following examples sets a maximum transition limit of 2.0 units on the specified library cell pins for the current mode for all corners and specific corner.

```
prompt> set_max_transition 2.0 [get_lib_pin slow/BUF/A]
prompt> set_max_transition 2.0 [get_lib_pin slow/BUF/A] -corner {c1}
```

The following example sets a maximum transition limit of 2.0 units on library cell pin for the current mode for specific corner for specified DB file name max.db. Only one max.db used in library preparation.

```
prompt> set_max_transition 2.0 -db max.db [get_lib_pin maxlib/BUF/A] -corner {c1}
```

The following example sets a maximum transition limit of 2.0 units on library cell pin for the current mode for specified full path name of DB files. Specifying full path name to uniquely identify max.db file.

```
prompt> set_max_transition 2.0 -db /top/a/b/max.db [get_lib_pin maxlib/BUF/A]
prompt> set_max_transition 2.0 -db /top/a/c/max.db [get_lib_pin maxlib/BUF/A]
```

SEE ALSO

current_design(2)
get_ports(2)

remove_max_transition(2)
report_cells(2)
report_design(2)
report_lib(2)
report_ports(2)

set_message_info

Set some information about diagnostic messages.

SYNTAX

string **set_message_info** -id message_id [-limit max_limit|-stop_on|-stop_ff]

string *message_id*
integer *max_limit*

ARGUMENTS

-id *message_id*

Information is to be set for the given *message_id*. The message must exist. Although different constraints allow different message types, no constraint allows severe or fatal messages.

-limit *max_limit*

Set the maximum number of occurrences for *message_id*. This is an integer greater than or equal to zero. If you set it to zero, that means the number of occurrences of the message is unlimited. Messages which occur after a limit is reached are automatically suppressed.

-stop_on

Force Tcl error if message is emitted.

-stop_off

Turn off a previous -stop_on directive

DESCRIPTION

The **set_message_info** command sets constraints on diagnostic messages (typically error, warning, and informational messages).

Currently, you can set a upper limit for the number of occurrences of a message. You can set this to zero to indicate that there is no limit. You can retrieve the current limit for a message using the **get_message_info** ***command. When the limit is exceeded, all future occurrences of the message are automatically suppressed. A count of total occurrences (including those suppressed) can be retrieved using get_message_info.***

EXAMPLES

The following example uses **set_message_info** to set a limit on the number of APP-027 messages to 100. When the 101st APP-027 message is about to be issued, you will be warned that the limit has been exceeded, and that all future occurrences will be suppressed.

```
prompt> set_message_info -id APP-027 -limit 100
prompt> do_command
Warning: can't find node U27.1 (APP-027)
Warning: can't find node U27.2 (APP-027)
Warning: can't find node U27.3 (APP-027)
 ...
Warning: can't find node U27.100 (APP-027)
Note - message 'APP-027' limit (100) exceeded.  Remainder will be suppressed.
1
```

SEE ALSO

get_message_info(2)
get_message_ids(2)
print_message_info(2)
suppress_message(2)

set_min_capacitance

Sets minimum capacitance for ports, library cell pins or designs.

SYNTAX

string **set_min_capacitance**
value
[-modes *mode_list*]
[-corners *corner_list*]
[-db *file_name*]
[-scenarios *scenario_list*]
object_list

Data Types

value float
mode_list list
corner_list list
file_name string
scenario_list list
object_list list

ARGUMENTS

value

Sets the capacitance limit. The value is greater than or equal to zero. This is the minimum total capacitance (pin plus wire capacitance) in library capacitance units.

-modes *mode_list*

Specifies the scenarios for which to apply the capacitance limit. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option must not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-corners *corner_list*

Specifies the scenarios for which to apply the capacitance limit. If this option is given, all scenarios of the given corners and the current mode will be used. The **-scenarios** option must not be given with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-db *file_name*

Specifies the full or partial DB file name that matches the unique DB file for library cell pins, to which to apply the capacitance limit. Option is valid only for library cell pin objects.

-scenarios *scenario_list*

Specifies the scenarios for which to apply the capacitance limit. The **-modes** or **-corners** option must not be given with this option.

object_list

Provides a list of input or inout ports, library cell pins or designs on which to set minimum capacitance.

DESCRIPTION

Specifies a minimum capacitance on input/inout ports, library cell pins or designs for the specified scenarios. If minimum capacitance is set on a port, the net connected to that port is expected to have total capacitance greater than the specified *capacitance_value*. If specified on a design, the default minimum capacitance for that design is set. Library cell pins can also have a **min_capacitance** value specified in the library.

If minimum capacitance is specified for library cell pin objects by using this command, the setting overrides the values specified in the library for the specified scenarios. The library cell pin limits are stored as design specific overrides in the scenarios of the current design and are only applicable to the current design.

The **-db** option is provided to specify library cell pin constraints for a specific DB library that was used to create the NDM library. The command doesn't access the file itself but uses the name to find matching panes to apply the constraints. The **report_lib** and **report_cells** commands can be used to find the DB file names. File_name can be a library name, a local DB file name or a full path name that matches a unique DB file. Constraints with **-db** files have higher precedence than without it. Use **remove_min_capacitance** to remove all library cell pin constraints including **-db** files.

The most restrictive of the design limit and the pin or port limit is used.

Minimum capacitance limits are managed on a per-scenario basis. For designs with multiple scenarios, you can specify different limits for different scenarios by using the **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the command applies to the current scenario. The tool issues an error if there is no current scenario. If the **-scenarios** option is specified, the command applies to all of the specified scenarios.

If the **-modes** option is specified, the command applies to all scenarios of the specified modes. The tool issues an error if none of the specified modes have any scenarios.

If the **-corners** option is specified, the command applies to all scenarios of the specified corners and the current mode. The tool issues an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are specified, the command applies to all scenarios of the specified corners and the specified modes. The tool issues an error if there are no such scenarios. It is an error to specify **-scenarios** with **-modes** or **-corners**.

The **report_ports -design_rule** command shows port minimum capacitance limits. The **report_design** command shows the default minimum capacitance setting for the scenarios of the current design.

To remove minimum capacitance limits from designs or ports, use **remove_min_capacitance**.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example sets a minimum capacitance limit of 0.2 units on ports "IN*".

```
prompt> set_min_capacitance 0.2 [get_ports "IN*"]
```

The following example sets the default minimum capacitance limit of 0.1 units on the current scenario of the current design.

```
prompt> set_min_capacitance 0.1 [current_design]
```

The following example sets a minimum capacitance limit on a library cell pin for the current mode for specified DB files.

```
prompt> set_min_capacitance 1.0 -db slow1v95.db [get_lib_pin slow/BUF/Y]
prompt> set_min_capacitance 1.1 -db slow1v40.db [get_lib_pin slow/BUF/Y]
```

SEE ALSO

current_design(2)
get_ports(2)
remove_min_capacitance(2)
report_design(2)
report_ports(2)
set_max_capacitance(2)

set_min_delay

Specifies a minimum delay for timing paths.

SYNTAX

status **set_min_delay**
[-rise]
[-fall]
[-reset_path]
[-ignore_clock_latency]
[-from *from_list*
| -rise_from *rise_from_list*
| -fall_from *fall_from_list*]
[-through *through_list*]
[-rise_through *rise_through_list*]
[-fall_through *fall_through_list*]
[-to *to_list*
| -rise_to *rise_to_list*
| -fall_to *fall_to_list*]
[-comment *comment*]
delay_value

Data Types

from_list list
rise_from_list list
fall_from_list list
through_list list
rise_through_list list
fall_through_list list
to_list list
rise_to_list list
fall_to_list list
comment string
delay_value float

ARGUMENTS

-rise

Specifies that the tool constrains only rising path delays. If neither **-rise** nor **-fall** is specified, both rising and falling delays are constrained.

-fall

Specifies that the tool constrains only falling path delays. If neither **-rise** nor **-fall** is specified, both rising and falling delays are constrained.

-reset_path

Removes existing point-to-point exception information from the specified paths. If used with **-to** only, all paths leading to the specified endpoints are reset. If used with **-from** only, all paths leading from the specified startpoints are reset. If used with **-from** and **-to**, only paths between those points are reset. Only information of the same rise or fall setup or hold type is reset. Using this option is equivalent to using the **reset_paths** command with similar arguments before issuing **set_min_delay**.

-ignore_clock_latency

Indicates that the launch and capture clock latencies is to be ignored when computing slack on the specified paths. Note that these paths will be considered as clock-less, and therefore, reporting these paths by their respective ignored clocks will result in unconstrained paths. To report these paths with ignore_clock_latency, the user may specify the launched and/or captured devices as options in the report_timing command.

-from *from_list*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If you specify a cell, one path startpoint on that cell is affected. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

Specifies a list of pins, ports, cells, and nets through which the paths must pass for maximum delay definition. Nets are interpreted to imply the leaf-level driver pins. If you omit **-through**, all timing paths specified using the **-from** and **-to** options are affected. You can specify **-through** more than one time in one command invocation. For a discussion of multiple **-through**, **rise_through**, and **fall_through** options, see the DESCRIPTION section.

-rise_through *rise_through_list*

This option is similar to the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than one time in a single command invocation. For a discussion of multiple **-through**, **rise_through**, and **fall_through** options, see the DESCRIPTION section.

-fall_through *fall_through_list*

This option is similar to the **-through** option, but applies only to paths with a falling transition at the specified objects. You can specify **-fall_through** more than one time in a single command invocation. For a discussion of multiple **-through**, **rise_through**, and **fall_through** options, see the DESCRIPTION section.

-to *to_list*

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-rise_to ***rise_to_list***

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to ***fall_to_list***

Same as the **-to** option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

delay_value

Specifies a floating point number that represents the required minimum delay value for specified paths. *delay_value* must have the same units as the technology library used during analysis. If a path startpoint is on a sequential device, clock skew is included in the computed delay. If a path startpoint has an input external delay specified, that delay value is added to the path delay. If a path endpoint is on a sequential device, clock skew and library setup time are included in the computed delay. If the endpoint has an output external delay specified, that delay is added into the path delay.

-comment ***comment***

Associates a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

DESCRIPTION

This command specifies a required minimum delay for timing paths in the current design. The path length for any startpoint in *from_list* to any endpoint in *to_list* must be greater *delay_value*.

Use **set_min_delay** in the following two cases:

1. To set a target minimum delay for output ports in a combinational design.

2. To override the default single cycle timing for paths, where **set_multicycle_path** is not sufficient.

The value of the **min_rise_delay** attribute cannot be greater than that of the **max_rise_delay** attribute for the same path (and similarly for fall attributes). If this condition occurs, the older attribute is removed.

Individual minimum delay targets are automatically derived from clock waveforms and port input or output delays. For more information, refer to the **create_clock**, **set_input_delay**, and **set_output_delay** command man pages.

The **set_min_delay** command is a point-to-point timing exception command; it overrides the default single-cycle timing relationship for one or more timing paths. Other point-to-point timing exception commands include **set_multicycle_path**, **set_max_delay**, and **set_false_path**. A **set_max_delay** or **set_min_delay** command overrides a **set_multicycle_path** command.

The more general commands apply to more than one path; either **-from** or **-to** is used (but not both), or clocks are used in the specification. Within a given point-to-point exception command, the more specific command overrides the more general. The following list of commands is arranged in order, from highest to lowest precedence (more specific to more general):

1. set_min_delay -from *pin* -to *pin*

2. set_min_delay -from *clock* -to *pin*

3. set_min_delay -from *pin* -to *clock*

4. set_min_delay -from *pin*

5. set_min_delay -to *pin*

6. set_min_delay -from *clock* -to *clock*

7. set_min_delay -from *clock*

8. set_min_delay -to *clock*

You can use multiple **-through**, **-rise_through**, and **fall_through** options in a single command to specify paths that traverse multiple points in the design. The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

```
-from A1 -through B1 -through C1 -to D1
```

If more than one object is specified within one **-through**, **-rise_through**, or **fall_through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
-from A1 -through {B1 B2} -through {C1 C2} -to D1
```

To remove information set by **set_min_delay**, use the **reset_paths** or **reset_design** command.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following command specifies that any delay path to port "Y" must be greater than 12.5 units.

```
prompt> set_min_delay 12.5 -to Y
```

The following command specifies that all paths from A1 and A2 to Z5 must have delays greater than 4.0 units.

```
prompt> set_min_delay 4.0 -from {A1 A2} -to Z5
```

The following example specifies that all timing paths from ff1/CP to ff2/D that pass through one or more of {U1/Z U2/Z} and one or more of {U3/Z U4/C} must have delays greater than 3.0 units.

```
prompt> set_min_delay 3.0 -from ff1/CP -through {U1/Z U2/Z} -through {U3/Z U4/C} -to ff2/D
```

The following example specifies that all timing paths from ff1/CP to ff2/D that rise through one or more of {U1/Z U2/Z} and fall through one or more of {U3/Z U4/C} must have delays greater than 3.0 units.

```
prompt> set_min_delay 3.0 -from ff1/CP -rise_through {U1/Z U2/Z} \
  -fall_through {U3/Z U4/C} -to ff2/D
```

SEE ALSO

current_design(2)
reset_design(2)

reset_paths(2)
set_false_path(2)
set_input_delay(2)
set_max_delay(2)
set_multicycle_path(2)
set_output_delay(2)

set_min_pulse_width

Sets a minimum pulse width constraint for clocks or clock pins.

SYNTAX

status **set_min_pulse_width**
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
[-low]
[-high]
value
object_list

Data Types

mode_list list
corner_list list
scenario_list list
value float
object_list list

ARGUMENTS

-modes *mode_list*

Specifies the scenarios on which to set the input delay value. If this option is given, all scenarios of the given modes are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-corners *corner_list*

Specifies the scenarios on which to set the input delay value. If this option is given, all scenarios of the given corners and the current mode is used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-scenarios *scenario_list*

Specifies the scenarios on which to set the input delay value. The **-modes** or **-corners** option must not be given with this option.

-low

Indicates that the minimum pulse width constraint specified by *value* applies only to low clock signal levels. If you do not specify the *-low* or *-high* option, both low and high pulses of clock signals are constrained.

-high

Indicates that the minimum pulse width constraint specified by *value* applies only to high clock signal levels. If you do not specify the *-low* or *-high* option, both low and high pulses of clock signals are constrained.

value

A positive floating-point value that specifies the minimum pulse width check to be applied to clock signals.

object_list

Specifies a list of clocks or clock pins in the current design to which the minimum pulse width check is applied. If you specify a clock, the constraint is applied to all clock pins connected to the clock.

DESCRIPTION

The **set_min_pulse_width** command specifies a minimum pulse width check to be applied to clock signals at clock pins of sequential devices. This value overrides the default values for clock tree minimum pulse width checks. Use this command for sequential clock pin pulse width checks to add a more restrictive value than the one specified in library. This setting is applied for current scenario.

To generate a report of pulse width constraints, use **report_min_pulse_width**. To remove minimum pulse width constraints set by **set_min_pulse_width**, use the **remove_min_pulse_width** command.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example sets a minimum pulse width requirement of 2.0 for both low and high pulses of clock signals.

```
pt_shell> set_min_pulse_width 2.0 [get_clocks CK1]
```

The following example sets a minimum pulse width requirement of 2.5 for the low pulse of the clock signal on the clock pin reg1/CK.

```
pt_shell> set_min_pulse_width -low 2.5 reg1/CK
```

SEE ALSO

remove_min_pulse_width(2)
report_min_pulse_width(2)

set_missing_via_check_options

Sets missing via check options for the RedHawk Analysis Fusion check missing via analysis flow. The settings are not saved in the IC Compiler II database, so you must be set the checking options in each session.

SYNTAX

status **set_missing_via_check_options**
[-redhawk_script_file *redhawk_script_file*]
[-exclude_stack_via]
[-check_valid_vias]
[-exclude_cell *exclude_cell_masters*]
[-exclude_inst *exclude_cell_instances*]
[-exclude_regions *regions*
[-sort_by_hot_inst]
[-threshold *voltage_value*]
[-min_width *min_width_value*]
[-pitch *pitch_value*]
[-ignore_inter_met]
[-toplayer *toplayer_name*]
[-bottomlayer *bottomlayer_name*]
[-gds]

Data Types

redhawk_script_file string
exclude_cell_masters list
exclude_cell_instances collection
regions list
voltage_value float
min_width_value float
pitch_value float
toplayer_name string
bottomlayer_name string

ARGUMENTS

-redhawk_script_file *redhawk_script_file*

Uses an existing RedHawk script file for missing via checking. You must source the user-given script file before exiting from RedHawk. This option allows you to reuse an existing RedHawk script which has filtering options specified in a previous run.

-exclude_stack_via

Excludes stack vias from missing via check.

-check_valid_vias

Reports if metal overlap can accommodate a valid via model.

-exclude_cell *exclude_cell_masters*

Excludes the area covered by all instances of the specified list of cells from missing via check.

-exclude_inst *exclude_cell_instances*

Excludes the area covered by the specified list of cell instances from missing via check.

-exclude_regions *regions*

Excludes the specified rectangular regions from missing via check. To specify a rectangular region, use the following format to specify the lower-left and upper-right corners of the rectangle:

{{*x1 y1*} {*x2 y2*}}

-sort_by_hot_inst

Sorts missing vias by voltage drop.

-threshold *voltage_value*

Specifies a voltage threshold that triggers a missing via check.

The default is 1mV. To disable voltage check filtering, set this option to -1.

-min_width *min_width_value*

Ignores segments with less than specified width.

-pitch *pitch_value*

Specifies pitch for missing via check on parallel wires crossing or touching GDS blocks.

-ignore_inter_met

Ignores wires on all layers between the -toplayer and the -bottomlayer.

-toplayer *toplayer_name*

Specifies the top layer

-bottomlayer *bottomlayer_name*

Specifies the bottom layer

-gds

Reports missing vias both inside the GDS block region and in routes cross over GDS blocks.

EXAMPLES

The following example configures settings for the current missing via check to report if metal overlap can accommodate a valid via model.

```
prompt> set_missing_via_check_options -check_valid_vias
```

SEE ALSO

analyze_rail(2)
report_missing_via_check_options(2)
remove_missing_via_check_options(2)
set_app_options(2)

set_msg

Configures messages and options to control tool output

SYNTAX

integer **set_msg**
[msgTag]
[-level *msgLevel*]
[-limit printLimit]
[-width lineWidth]
[-reset_to_default]
[-only_level msgLevel]
[-add_filter pattern]
[-remove_filter pattern]

Data Types

msgLevel 'off', 'debug', 'verbose', 'info', 'warning', 'error', 'serror' or 'fatal'
printLimit uint
lineWidth uint
pattern string

ARGUMENTS

msgTag

Tag pattern to which the setting applies. This can be a specific message tag ('ABC-1234') or use wildcards to match arbitrary sets of messages. E.g. 'ZRT-*' matches all zroute messages.

-level *msgLevel*

Sets the level of the matching message(s). To stop a message from printing (that is suppress the message) set it level to 'off'. Setting a message whose level is verbose to 'info' or higher causes it to become printable. A message level of 'verbose' and below does not print. If message level is set to 'serror' (Severe_error), execution will stop at the end of the TCL command the produced the message. A message level 'fatal' will cause the tool to crash when the error occurs.

-limit *printLimit*

Sets the maximum number of times that a message may be printed per TCL file. A value of 0 means no print restrictions. The limit applies per TCL file. At the end of a TCL file a summary of the messages that overstepped their limit will be printed.

-width *lineWidth*

Causes message and all other tool output to wrap when line width exceeds the given value. The default is unlimited.

-reset_to_default

Resets the message level back to its default setting.

-only_level *msgLevel*

Applies setting only to messages of the indicated level.

-add_filter *pattern*

Causes all (non-message) output that matches 'pattern' to be suppressed. This is useful to remove garbage output from overflowing the log file. The filter only applies to non-message output that does not have a tag.

-remove_filter *pattern*

Removes a previously set filter

RETURN VALUE

1 upon success

DESCRIPTION

The **set_msg** command changes the properties of existing messages or other tool output. Use this to turn off the printing of tedious tool output, or - conversely - to cause the tool to print extra verbose output.

The tool contains many thousands of built-in messages, each identified by unique 'tag' in the form 'ABC-123' or 'ABC-1234'. Use 'report_msg' to find and list the available messages, or to list the messages that were printed until now. Each message has a default level:
off The message is suppressed and will not print. By default no messages are at level 'off'
debug Internal debug message that does not print by default. Only accessible by Synopsys developers
verbose Additional information that does not print by default
info Generic information output. This message prints by default
warning Warning message: abnormal issue that warrants attention but is not show-stopping
error Error message: an abnormality that must be addressed by the user
serror Severe_Error message: a show-stopping error. This will stop flow execution at the end of the TCL command.
fatal Deadly error, causing the tool to crash immediately

Using *set_msg -level* the severity ('level') of any message can be changed: downgraded to non-printing, or upgraded to printing or worse. To suppress a specific message: look up its tag and switch it off like this:

```
prompt> set_msg ABC-1234 -level off
```

It is also possible to set a limit on the number of times a message is printed using '-limit'. The tool meticulously registers the occurrence of a message, whether printing on not. Using 'get_msg' it is easy to detect whether a particular message occurred. This is a clean alternative to searching the log file, because the user can take conditional action when the message occurred (printed or not)
In addition to controlling what messages are printed or suppressed, *set_msg -filter* allows the user to suppress unwanted tool output that is not tagged as a message. There are many tens of thousands of those in the tool.

EXAMPLES

Suppress the printing of a specific noisy message:

```
set_msg ABC-2001 -level off
```

Upgrade a specific 'verbose' message CVG-2001 such that it will be printed:

```
set_msg CVG-2001 -level info
```

Upgrade all verbose messages of the FLW package such that they are printed:

```
set_msg FLW* -only_level verbose -level info
```

Upgrade a specific set of (debug) messages:

```
set_msg FLW*52* -level info
```

Upgrade all Verilog warning messages to 'Serious_error', such that the flow stops:

```
set_msg VER* -only_level warning -level serror
```

Suppress a specific Error message by downgrading it to a warning:

```
set_msg ZRT-510 -level warning
```

Downgrade all VER* error messages to a warning. So this is effectively a continue_on_error but only for specific Verilog errors.

```
set_msg VER* -only_level error -level warning
```

Reset the level of all messages back to default

```
set_msg * -reset_to_default
```

Limit the number of times the message is printed to 4

```
set_msg ABC-123 -limit 4
```

Remove the print limit by setting it back to 0

```
set_msg ABC-123 -limit 0
```

Remove all print limits on any message

```
set_msg * -limit 0
```

Suppress any tool output that contains the string 'processing module'

```
set_msg -filter "processing module"
```

SEE ALSO

report_msg(2)
get_msg(2)

set_multi_input_switching_coefficient

Sets delay coefficients for multi-input switching (MIS) analysis for a specified list of library cells.

SYNTAX

int **set_multi_input_switching_coefficient**
[-rise]
[-fall]
-delay *coeff_value*
object_list

Data Types

coeff_value float
object_list list

ARGUMENTS

-rise

Applies the coefficients to rise transitions only.

-fall

Applies the coefficients to fall transitions only.

-delay *coeff_value*

Indicates the base coefficient of the cell delay for multi-input switching analysis when the specified library cell is used.

object_list

Specifies a list of library cells for multi-input switching analysis.

DESCRIPTION

This command sets the base coefficients for the cell delay during multi-input switching (MIS) analysis. To enable multi-input switching analysis, set the application option **time.enable_multi_input_switching_analysis** to true; the default of this application option is false.

If the application option **time.enable_multi_input_switching_timing_window_filter** is set to true, the tool uses the base coefficient value to derive the actual MIS factor by considering the arrival windows of the input pins and their overlap analysis.

If the application option **time.enable_multi_input_switching_timing_window_filter** is set to false, the base coefficient value is directly applied to the MIS factor. The default of application option **time.enable_multi_input_switching_timing_window_filter** is true.

If you use the **set_multi_input_switching_coefficient** command multiple times on the same library cell, the last specified coefficient overwrites the previously specified coefficients.

To apply MIS coefficient to library cell in one specific pane, specify the db file name together with the library cell using **get_lib_cell**, otherwise it's applied to all panes. For the same library cell, the pane specific setting will override global setting.

In scaling flow, if the applied PVT matches one of the source db in the scaling group, then that source db is checked for pane specific setting first, if not found we will check if there is global setting, if not, we will check the next best matching db in the scaling group.

EXAMPLES

The following example sets the delay coefficient value of 0.7 for all instances of library cell AND2 in the library MY_LIB:

```
prompt> set_multi_input_switching_coefficient -delay 0.7 [get_lib_cells MY_LIB/AND2]
```

The following example sets the delay coefficient value of 0.8 for all instances of library cell AND2, and the PVT matches MY_LIB.db:

```
prompt> set_multi_input_switching_coefficient -delay 0.8 [get_lib_cells MY_LIB.db:MY_LIB/AND2]
```

SEE ALSO

report_multi_input_switching_coefficient(2)
reset_multi_input_switching_coefficient(2)
enable_multi_input_switching_analysis(3)
enable_multi_input_switching_timing_window_filter(3)

set_multi_vth_constraint

Enable percentage vth flow, set low vth percentage limit and cost type

SYNTAX

set_multi_vth_constraint
-low_vt_percentage *percentage_limit*
[-cost *cost_type*]

Data Types

percentage_limit float: in range [0.0, 100.0]
cost_type string: cell_count or area

ARGUMENTS

-low_vt_percentage *percentage_limit*

Specifies low vth percentage limit based on cost type. This is a required option. Default value is 100.0.

-cost *cost_type*

Specifies how low vth percentage limit is calculated. The low vth percentage can be based on cell count or cell area. Default value is cell_count.

DESCRIPTION

This command enables percentage vth flow and sets percentage vth constraints, including low vth percentage limit and cost type. These settings are persistent across shell sessions.

If percentage vth flow is enabled, optimization commands including **compile_fusion**, **place_opt**, **clock_opt** and **route_opt** will honor the low vth percentage limit based on cost type.

PREREQUISITES

Low_vt percentage constraint needs to be provided, and at least one threshold voltage group needs to be assigned to low_vt type. For example:

prompt> **set_attribute [get_lib_cells *lvt*/*] threshold_voltage_group LVT**
prompt> **set_threshold_voltage_group_type -type low_vt LVT**

EXAMPLES

```
prompt> set_multi_vth_constraint -low_vt_percentage 10 -cost area
Percentage vth flow  : enabled
Percentage vth limit : 10.000 (%)
Percentage vth cost  : area
```

SEE ALSO

reset_multi_vth_constraint(2)
report_multi_vth_constraint(2)
report_threshold_voltage_group(2)
set_vth_threshold_controls(2)
set_threshold_voltage_group_type(2)

set_multibit_options

Specifies the options to use for multibit cell mapping during compile.

SYNTAX

status **set_multibit_options**
[-exclude *object_list*]
[-stage *stage_name*]
[-bus_registers_only]
[-name_prefix *prefix_string*]
[-name_concatenation_character *concatenate_string*]
[-input_map_file *file_name*]
[-exclude_library_cells *library_cells*]
[-slack_threshold *slack_value*]
[-verbose_file *file_name*]

ARGUMENTS

-exclude *object_list*

Specifies the netlist objects to exclude from multibit banking during compile. The *object_list* might contain collections of individual cells, buses, or modules. For buses, all the contained scalar elements are excluded from multibit banking. For modules, all relevant cells in the top-level hierarchy are excluded from multibit banking.

This option can combine with "-stage" to specify at which stage multibit exclusion will be effective (see "-stage" option below).

-stage *stage_name*

Specifies the stage that "-exclude" option should be applied to. Valid stage names are *rtl*, *physical*, and *all*.

This "-stage" option can only be combined with "-exclude" option; it cannot be combined with any other options of set_multibit_options.

If "-stage" option is absent, the behavior will be identical to "-stage all".

-bus_registers_only

Specifies that cells should be banked with other cells of same bus only.

-name_prefix *prefix_string*

Specifies the string to prefix the instance names created from multibit banking.

-name_concatenation_character *concatenate_string*

Specifies the string to be used to concatenate the names of the instances created from multibit banking.

-input_map_file *file_name*

Specifies the name of the input map file that contains functional group information and mapping information about which multibit register bank replaces which single-bit registers for each functional group.

If this option is not specified, the tool identifies the single-bit registers that can be replaced by available multibit registers in the libraries based on the functional information of the library cells; the tool then uses that information to control mapping.

The format for the input map file is described in the FILE FORMATS section of this man page.

-exclude_library_cells *library_cells*

Specifies the list of library cells whose instances are excluded from multibit banking.

-slack_threshold *slack_value*

Specifies the slack value for a register to be considered for multibit banking. Registers having slack value below this number are ignored from multibit banking. By default, the tool considers all registers for multibit banking.

-verbose_file *file_name*

Specifies the file name in which verbose messages and Summary related to banking is dumped.

This file consists of information related to cells ignored during banking flow and it's reason. It also contains information regarding names of cells getting banked with their corresponding newly created cell names.

This command option will only work with physical aware multibit banking and not with RTL banking.

FILE FORMATS

Format for specifying -input_map_file

You can specify the input map file sequence as shown:

reg_group_name {*list_of_reference_of_single_bit_flops* }
bits { *number_of_instances ref_multibit_flop* } { ... }
bits { *number_of_instances ref_multibit_flop* } { ... }

Where *reg_group_name* specifies the name of the register group, *list_of_reference_of_single_bit_flops* specifies the list of references of single-bit registers. **bits** specifies the number of single-bit registers in a group to be replaced, *number_of_instance* specifies the number of multibit registers to be used, and *ref_multibit_flop* specifies the reference multibit library cell to be used.

For example:

```
reg_group_1 {REGX1 REGX2 REGX4}
2 {1 MREG2}
3 {1 MREG2}
4 {1 MREG4}
5 {1 MREG4}
6 {1 MREG2} {1 MREG4}
reg_group_2 {REGNX1 REGNX2 REGNX4}
2 {1 MREG2N}
3 {1 MREG2N}
4 {1 MREG4N}
5 {1 MREG4N}
6 {1 MREG2N} {1 MREG4N}
```

reg_group_1 {REGX1 REGX2 REGX4} means that single-bit registers whose reference is either REGX1, REGX2, or REGX4 can be grouped together. **4 {1 MREG4}** specifies that a group of four single-bit registers can be replaced by one cell whose reference is MREG4. Similarly, **6 {1 MREG2} {1 MREG4}** means a group of six single-bit registers can be replaced using one instances of MREG2 and one instance of MREG4.

EXAMPLES

The following example excludes cell reg[1] from multibit banking:

prompt> **set_multibit_options -exclude [get_cells "reg[1]"]**

The following example excludes the reg bus from multibit banking:

prompt> **set_multibit_options -exclude [get_cell_buses "reg"]**

The following example excludes the reg bus from RTL multibit banking (but "reg" will not be excluded from physical multibit banking or any other types of multibit banking):

prompt> **set_multibit_options -exclude [get_cell_buses "reg"] -stage rtl**

The following example excludes all sequential cells of the mod subdesign in the top-level hierarchy from multibit banking:

prompt> **set_multibit_options -exclude [get_modules "mod"]**

The following example queries cells that are excluded from multibit banking using the **set_multibit_options** command:

prompt> **get_attribute -objects [get_cells {reg[1]}] -name exclude_multibit**

SEE ALSO

compile.multibit.enable(3)
compile(2)
report_cell_buses(2)
report_multibit_banking(2)
get_attribute(2)

set_multicycle_path

Defines the multicycle path.

SYNTAX

status **set_multicycle_path**
[-setup] [-hold]
[-rise] [-fall]
[-start] [-end]
[-reset_path]
[-from *from_list*
| -rise_from *rise_from_list*
| -fall_from *fall_from_list*]
[-through *through_list*]*
[-rise_through *rise_through_list*]*
[-fall_through *fall_through_list*]*
[-to *to_list*
| -rise_to *rise_to_list*
| -fall_to *fall_to_list*]
[-comment *comment*]
path_multiplier

Data Types

from_list list
rise_from_list list
fall_from_list list
through_list list
rise_through_list list
fall_through_list list
to_list list
rise_to_list list
fall_to_list list
comment string
path_multiplier int

ARGUMENTS

-setup

Uses the specified *path_multiplier* for setup (maximum delay) calculations. Note that changing the *path_multiplier* for setup also affects the default hold check. If neither **-setup** nor **-hold** is specified, *path_multiplier* is used for setup calculations and 0 is used for hold calculations.

-hold

Uses the specified *path_multiplier* for hold (minimum delay) calculations. Note that changing the *path_multiplier* for setup also affects the default hold check. If neither **-setup** nor **-hold** is specified, *path_multiplier* is used for setup calculations and 0 is used for hold calculations.

-rise

Uses *path_multiplier* only for rising path delays. If neither **-rise** nor **-fall** is specified, both rising and falling delays are affected. Rise refers to a rising value at the path endpoint.

-fall

Uses *path_multiplier* only for falling path delays. If neither **-rise** nor **-fall** is specified, both rising and falling delays are affected. Fall refers to a falling value at the path endpoint.

-start

Specifies that the multicycle information is relative to the period of the start clock. The **-start** and **-end** options are needed only for multifrequency designs; otherwise, start and end are equivalent. The start clock is the clock source related to the register or primary input at the path startpoint.

The default is to move the setup check relative to the end clock, and the hold check relative to the start clock. A setup multiplier of 2 with **-start** moves the relation backward one cycle of the start clock. A hold multiplier of 1 with **-start** moves the relation forward one cycle of the start clock.

-end

Specifies that the multicycle information is relative to the period of the end clock. The **-start** and **-end** options are needed only for multifrequency designs; otherwise, start and end are equivalent. The end clock is the clock source related to the register or primary output at the path endpoint.

The default is to move the setup check relative to the end clock, and the hold check relative to the start clock. A setup multiplier of 2 with **-end** moves the relation forward one cycle of the end clock. A hold multiplier of 1 with **-end** moves the relation backward one cycle of the end clock.

-reset_path

Removes the existing point-to-point exception information from the specified paths. If used with **-to** only, all paths leading to the specified endpoints are reset. If used with **-from** only, all paths leading from the specified startpoints are reset. If used with **-from** and **-to**, only paths between those points are reset. Only information of the same rise/fall setup/hold type is reset. Using this option is equivalent to using the **reset_paths** command with similar arguments before issuing **set_multicycle_path**.

-from *from_list*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If a cell is specified, one path startpoint on that cell is affected. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

Specifies a list of pins, ports, cells and nets through which the multicycle paths must pass. Nets are interpreted to imply the leaf-level driver pins. If you omit **-through**, all timing paths specified using the **-from** and **-to** options are affected. You can specify **-through** multiple times in a single command invocation. For a discussion of multiple **-through**, **rise_through**, and **fall_through** options, see the DESCRIPTION section.

-rise_through *rise_through_list*

This option is similar to the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** multiple times in a single command invocation. For a discussion of multiple **-through**, **rise_through**, and **fall_through** options, see the DESCRIPTION section.

-fall_through *fall_through_list*

This option is similar to the **-through** option, but applies only to paths with a fall transition at the specified objects. You can specify **-fall_through** multiple times in a single command invocation. For a discussion of multiple **-through**, **rise_through**, and **fall_through** options, see the DESCRIPTION section.

-to *to_list*

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-comment *comment*

Specifies a user-defined comment.

path_multiplier

Specifies the integer number of cycles the data path must have for setup or hold, relative to the startpoint or endpoint clock, before data is required at the endpoint. For example, specifying a *path_multiplier* of 2 for setup implies a 2-cycle data path. If you use **-setup**, *path_multiplier* is applied to setup path calculations. If you use **-hold**, *path_multiplier* is applied to hold path calculations. If you do not specify either **-setup** or **-hold**, *path_multiplier* is used for setup and 0 is used for hold. Note that changing the multiplier for setup affects the hold check as well.

-comment *comment*

Associate a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

DESCRIPTION

This command specifies the number of cycles that the data path must have for setup or hold, so that designated timing paths in the current design have no default setup or hold relations.

Single cycle timing relationships are determined for paths between clocked elements; the rules are based on active edges. For flip-flops, a single active edge both launches and captures data. For latches, the open edge launches data and the close edge latches data.

The setup check ensures that the correct data signal is available on destination registers in time to be correctly latched. The default setup behavior, called single-cycle setup, is as follows:

For multifrequency designs, there can be multiple setup relations between two clocks. For every latch edge of the destination clock, the setup check determines the nearest launch edge that precedes each latch edge. The smallest value of (setup_latch_edge - setup_launch_edge) determines the maximum delay requirement for this path.

You can override this default relationship by applying **set_multicycle_path** or **set_max_delay** to clocks, pins, ports, or cells. For example, setting the setup path multiplier to 2 with the **set_multicycle_path** command delays the latch edge one clock pulse. Note that changing the setup multiplier also affects the default hold check.

Most often, the setup check moves relative to the end clock. This move changes the data latch time at the path endpoint. In multi-frequency designs, use **set_multicycle_path -setup -start** to move the data launch time backward. The hold check ensures that data from the source clock edge that follows the setup launch edge is not latched by the setup latch edge, and also ensures that data from the setup launch edge is not latched by the destination clock edge that precedes the setup latch edge.

The hold check is determined relative to each valid setup relationship, after applying multicycle path multipliers. The most restrictive (largest) difference of (hold_latch_edge - hold_launch_edge) is used as a minimum delay requirement for the path.

Important Note: The hold relation is conservative. In some cases you need to override the default hold relation. For multifrequency designs, it is more straightforward to use the **set_min_delay** command to override the default instead of using **set_multicycle_path -hold**.

There are separate setup and hold multipliers for rise and fall. In most cases, these are set to the same value. You can use the **-rise** or **-fall** option to apply different values.

The **set_multicycle_path** command is a point-to-point timing exception command. The command can override the default single-cycle timing relationship for one or more timing paths. Other point-to-point timing exception commands include **set_max_delay**, **set_min_delay**, and **set_false_path**. False path information always takes precedence over multicycle path information. A specific **set_max_delay** or **set_min_delay** command overrides a general **set_multicycle_path** command.

The more general commands apply to more than one path; either **-from** or **-to** (but not both), or clocks are used in the specification. Within a given point-to-point exception command, the more specific command overrides the more general. The following list shows commands from highest to lowest precedence (more specific to more general):

1. set_multicycle_path -from *pin* -to *pin*
2. set_multicycle_path -from *pin* -to *clock*
3. set_multicycle_path -from *pin*
4. set_multicycle_path -from *clock* -to *pin*
5. set_multicycle_path -to *pin*
6. set_multicycle_path -from *clock* -to *clock*
7. set_multicycle_path -from *clock*
8. set_multicycle_path -to *clock*

You can use multiple **-through**, **-rise_through**, and **-fall_through** options in a single command to specify paths that traverse multiple points in the design. The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

```
-from A1 -through B1 -through C1 -to D1
```

If more than one object is specified within one **-through**, **-rise_through**, or **fall_through** option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
-from A1 -through {B1 B2} -through {C1 C2} -to D1
```

To undo a **set_multicycle_path** command, use the **reset_paths** command. The **reset_design** command removes all constraints and user settings from the design, including those set by **set_multicycle_path**.

To disable setup and hold calculations for paths, use **set_false_path**. To list the exceptions on a design, use **report_exceptions**.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example sets all paths between latch1b and latch2d to 2-cycle paths for setup. Hold is measured at the previous edge of the clock at latch2d.

```
prompt> set_multicycle_path 2 -from { latch1b } -to { latch2d }
```

The following example moves the hold check to the preceding edge of the start clock.

```
prompt> set_multicycle_path -1 -from { latch1b } -to { latch2d }
```

The following example is a two-phase level-sensitive design, where the designer expects paths from phi1 to phi1 to be zero cycles.

```
prompt> set_multicycle_path 0 -from phi1 -to phi1
```

The following example uses **-start** to specify a 3-cycle path relative to the clock at the path startpoint. Clock sources are specified, affecting all sequential elements clocked by that clock, or ports with input or output delay relative to that clock.

```
prompt> set_multicycle_path 3 -start -from { clk50mhz } -to { clk10mhz }
```

The following example sets all rising paths to ff12/D to 2 cycles, and falling paths to 1 cycle.

```
prompt> set_multicycle_path 2 -rise -to { ff12/D }
prompt> set_multicycle_path 1 -fall -to { ff12/D }
```

The following multifrequency example shows a 20ns clock to 10ns clock with multicycle path of 2. Assume a path from ff1 (clocked by CLK2) to ff2 (clocked by CLK1).

```
prompt> create_clock -period 20 -waveform {0 10} CLK2
prompt> create_clock -period 10 -waveform {0 5} CLK1
prompt> set_multicycle_path 2 -setup -from ff1/CP -to ff2/D
```

The single-cycle setup relation is from the CLK1 edge at 0ns to the CLK2 edge at 10ns. With the multicycle path, the setup relation is now 20ns (from CLK1 edge at 0ns to CLK2 edge at 20ns). The hold relations are determined according to that setup relation.

1. Data from the source clock edge that follows the setup launch edge must not be latched by the setup latch edge. This implies a hold relation of 0ns (from CLK1 edge at 20ns to CLK2 edge at 20ns).

2. Data from the setup launch edge must not be latched by the destination clock edge that precedes the setup latch edge. This implies a hold relation of 10ns (from CLK1 edge at 0ns to CLK2 edge at 10ns).

The most restrictive (largest) hold relation is used, so the minimum delay requirement for this path is 10ns. This is a conservative check which might not apply to certain designs. Often you know that the destination register is disabled for the clock edge at 10ns. You can modify this default hold relation with the following to get an 0ns requirement.

```
prompt> set_multicycle_path 1 -hold -end -from ff1/CP -to ff2/D
```

However, a simpler approach is to use the following:

```
prompt> set_min_delay 0 -from ff1/CP -to ff2/D
```

The following example sets all timing paths from ff1/CP to ff2/D which passe through one or more of {U1/Z U2/Z} and one or more of {U3/Z U4/C} to 2 cycle paths for setup.

```
prompt> set_multicycle_path 2 -from ff1/CP -through {U1/Z U2/Z} \
  -through {U3/Z U4/C} -to ff2/D
```

SEE ALSO

current_design(2)
report_exceptions(2)
reset_design(2)
reset_paths(2)
set_false_path(2)
set_max_delay(2)
set_min_delay(2)

set_multisource_clock_subtree_constraints

Defines constraints for structural synthesis of multisource clock tree synthesis (CTS) subtrees.

SYNTAX

status **set_multisource_clock_subtree_constraints**
[-name *name*]
-clock *clock_name*
[-cells *cell_list*]
[-pins *pins_or_ports*]
[-ignore_for_icg_reordering]
[-target_level *levels*]

Data Types

name string
clock_name string or collection
cell_list collection of cells
pins_or_ports list or collection
levels integer

ARGUMENTS

-name *name*

Specifies the structural multisource subtree set to be constrained. The **-name** option is optional. It allows you to reference multiple subtree sets per clock. Without this option, the set is named as the clock object. Names must be unique for all defined set of subtrees.

-clock *clock_name*

Specifies a clock which will be balanced for the specified set of subtrees. This option is required.

-cells *cell_list*

Specifies a collection of cells on which constraints such as "ignore for ICG reordering" will be applied. The constraints on the cells are honored by the **synthesize_multisource_clock_subtrees** command.

-pins *pins_or_ports*

Specifies a collection of pins on which constraints such as "target level" will be applied. The pins must be the sinks of the clock tree. The constraints on the pins are honored by the **synthesize_multisource_clock_subtrees** command.

-ignore_for_icg_reordering

Ignores cells specified by the **-cells** option during the integrated clock gating (ICG) reordering step for the

synthesize_multisource_clock_subtrees command. This option can only be used together with the **-cells** option.

-target_level ***levels***

Defines the exact number of levels in the **-pins** option to synthesize with the **synthesize_multisource_clock_subtrees** command. This option can only be used together with the **-pins** option. The minimum level value is 0, the maximum level is restricted to 100. A target level of 0 or level less than the existing level would mean the level would be maintained.

DESCRIPTION

The **set_multisource_clock_subtree_constraints** command applies synthesis constraints on cells or pins for structural multisource CTS synthesis with the **synthesize_multisource_clock_subtrees** command.

This command extends the **set_multisource_clock_subtree_options** command for setting cell- or pin-specific option or constraints.

To better control subtree logic level balancing, specify the **-target_level** option. To better control subtree synthesis transforms for ICG reordering, specify the **-ignore_for_icg_reordering** option.

If a setting has already been applied for a subtree set and is reapplied for the same set, then the new setting is additive to the existing list.

Use the **remove_multisource_clock_subtree_constraints** command to clear constraints set with the **set_multisource_clock_subtree_constraints** command. Use the **report_multisource_clock_subtree_constraints** command to report constraints set with the **set_multisource_clock_subtree_options** command. Use the **synthesize_multisource_clock_subtrees** command to synthesize the subtrees that are constrained by the **set_multisource_clock_subtree_constraints** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This example specifies a target level for a pin.

```
prompt> set_multisource_clock_subtree_constraints \
  -clock [get_clock clk] -pins [get_pins {reg1/ck reg2/ck}] \
  -target_level 5
```

This example ignores ICG cells during ICG reordering in subtree synthesis:

```
prompt> set_multisource_clock_subtree_constraints \
  -clock [get_clock clk] -cells [get_cells {icg1 icg2}] \
  -ignore_for_icg_reordering
```

SEE ALSO

remove_multisource_clock_subtree_constraints(2)
remove_multisource_clock_subtree_options(2)
report_multisource_clock_subtree_constraints(2)
report_multisource_clock_subtree_options(2)
set_multisource_clock_subtree_options(2)
synthesize_multisource_clock_subtrees(2)

set_multisource_clock_subtree_options

Defines constraints for structural synthesis of multisource clock tree synthesis (CTS) subtrees.

SYNTAX

status **set_multisource_clock_subtree_options**
[-name *name*]
-clock *clock_name*
-driver_objects *pins_or_ports_or_nets*
[-max_total_wire_delay *wire_delay*]
[-corners *corner_list*]
[-dont_merge_cells *cell_list*]
[-balance_levels true | false]
[-target_level *levels*]
[-enable_icg_reordering true | false]

Data Types

name string
clock_name collection
pins_or_ports_or_nets collection
wire_delay float
corner_list collection
cell_list collection
levels integer

ARGUMENTS

-name *name*

Specifies the structural multisource subtree to constrain. During synthesis, sinks and clock logic can be assigned to any subtree within their set of driver objects, and all sinks in that set will be balanced together.

The **-name** option is optional. It allows the user to define multiple subtree sets per clock. Without this option the set is named as the clock object. Names must be unique for all defined set of subtrees.

-clock *clock_name*

Specifies a clock to balance for the specified set of subtrees. This option is required. All sinks reachable from the driver objects are balanced for only one clock.

If sinks in the set do not receive the specified clock, then an error will be issued when running the **synthesize_multisource_clock_subtrees** command.

-driver_objects *pins_or_ports_or_nets*

Specifies driver objects whose fanouts and sinks can be redistributed. This option is required. The driver object can be a standard cell output pin, a clock input port, or a net. If a net name is used, the driver pin of that net is inferred as the driver object.

For example, if clock buffers are driven by the clock mesh directly, then the output pins of these buffers can be specified as the driver objects. Alternatively, if a top-level mesh is pushed down to the block-level and the mesh straps act as the drivers themselves, the ports for each mesh strap can be specified as driver objects.

Each driver object should have a common clock propagating through it, because every subtree in a set is synthesized for one clock only.

-max_total_wire_delay ***wire_delay***

Specifies the maximum wire delay from any subtree driver to any sink reachable form the driver objects.

This value is an absolute amount of wire delay on the clock paths from the subtree driver to the sink, not a percentage. The wire delay is specified in time units as configured with the **set_user_units** command. By default, time is specified in nanoseconds.

If **-max_total_wire_delay** is specified, then **-corners** must also be specified to determine the corner or corners to which the total wire delay setting applies.

-corners ***corner_list***

Specifies the corner or corners to which the max_total_wire_delay constraint applies. This option has no meaning when paired with other options of this command. If **-max_total_wire_delay** is used, then corners is a required option.

-dont_merge_cells ***cell_list***

Specifies a collection of cells which should not be merged with other cells by the **synthesize_multisource_clock_subtrees** command.

-balance_levels true | false

Enables or disables level balancing. The default is false and level balancing is disabled.

-target_level ***levels***

Defines the number of levels to the sinks which the command **synthesize_multisource_clock_subtrees** should synthesize. This option can only be used together with the **-balance_levels** option. The minimum level value is 2 while the maximum is restricted to 100.

-enable_icg_reordering true | false

Reorders ICG cells with a driving buffer.

DESCRIPTION

The **set_multisource_clock_subtree_options** command applies synthesis constraints for structural multisource CTS synthesis using the **synthesize_multisource_clock_subtrees** command.

Typically, a multisource CTS approach involves some mesh, H-tree, or other global clock distribution structure that drives a collection of driver cells. Sometimes, there are no explicit driver cells, and the mesh straps themselves act as the driver objects. In either case, the clock trees after the driver objects are called subtrees, and this command is used to specify the implementation requirements on those subtrees.

Each subtree is driven by a unique driver object. Multiple subtrees can be grouped together for skew balancing and sink assignment across those subtrees.

This command performs four main functions:

- Defines set of subtrees for balancing and sink assignment
- Defines which clock to balance in each set
- Defines structural synthesis constraints for each set
- Defines the number of levels at which the sinks should be synthesized

First, subtree sets are defined by specifying their driver objects. These subtrees are balanced together during structural synthesis. The subtrees are also considered equivalent to each other when reassigning sinks to different subtrees. Any sink of any subtree in a set can be reassigned to another subtree of that set, based on the placement of the sinks and the driver objects.

The subtrees that belong to a set are defined by their driver objects as specified with the **-driver_objects** option. Each subtree has a single driver object. Driver objects are typically either the output pins of buffers attached to the mesh, or are clock input ports associated with pushed-down mesh topologies from the top-level design. In general, a driver object can be any clock cell output pin, clock input port, or clock net. If a clock net is used, then the driver pin of that net is considered the driver object.

With the **-driver_objects** and **-clock** options, subtrees should now be completely assigned into sets and the correct clock specified for balancing in each of those groups. Now constraints for each set can be defined.

You can define the maximum amount of wire delay on the path from any subtree driver to the sinks of that set of driver objects. This is specified with the **-max_total_wire_delay** option. The max_total_wire_delay setting is a time value specifying an absolute maximum amount of wire delay for each clock path. The **-corners** option is required if specifying **-max_total_wire_delay**, and is used to constrain for which corner or corners that maximum total wire delay value applies.

You can optionally enable level balancing of the subtrees. If a target level is not given by this command, the subtrees are buffered such that all sinks are at the same level, i.e. have the same number of clock cells on the path to its driver object. Use the **-target_level** option to explicitly define the number of levels.

If a particular setting has already been applied for a subtree set, and then it is applied again for the same set, then the original setting is overwritten by the new setting.

Use the **remove_multisource_clock_subtree_options** command to clear constraints that have been set with this command. Use the **report_multisource_clock_subtree_options** command to report constraints set with this command. Use the **synthesize_multisource_clock_subtrees** command to synthesize the subtrees, as constrained by this command.

EXAMPLES

This example assigns subtree driver objects to a clock.

```
prompt> set_multisource_clock_subtree_options -clock [get_clock clk] \
  -driver_objects [get_pins {driver1/Z driver2/Z}]
prompt> set_multisource_clock_subtree_options -clock [get_clock clk] \
  -max_total_wire_delay 0.300 -corners [get_corners {c1 c2}]
```

This example performs the exact same function as the two commands in the previous example. It illustrates that it's not necessary to split the constraints into separate command calls. All constraints are associated to the set, whether they are specified together or separately.

```
prompt> set_multisource_clock_subtree_options \
  -driver_objects [get_pins {driver1/Z driver2/Z}] \
  -clock [get_clocks clk] \
  -max_total_wire_delay 0.300 -corners [get_corners {c1 c2}]
```

This example shows an incorrect way to specify multiple driver objects for a given option set, and the correct way to specify the command. Each option can be applied to a clock one time; if called a second time, then the second setting overwrites the first.

```
## Incorrect second constraint overwrites the first
prompt> set_multisource_clock_subtree_options -clock clk \
  -driver_objects [get_pins driver1/Z]
prompt> set_multisource_clock_subtree_options -clock clk \
  -driver_objects [get_pins driver2/Z]
## Correct way to specify both driver objects together for the group
prompt> set_multisource_clock_subtree_options -clock clk \
  -driver_objects [get_pins {driver1/Z driver2/Z}]
```

This example excludes two cells from getting merged.

```
prompt> set_multisource_clock_subtree_options \
  -driver_objects [get_pins {driver1/Z driver2/Z}] \
  -clock [get_clocks clk] \
  -dont_merge_cells [get_cells {u1 u2}]
```

This example enables automatic level balancing.

```
prompt> set_multisource_clock_subtree_options \
  -driver_objects [get_pins {driver1/Z driver2/Z}] \
  -clock [get_clocks clk] \
  -balance_levels true
```

This example enables automatic level balancing with an explicit number of levels.

```
prompt> set_multisource_clock_subtree_options \
  -driver_objects [get_pins {driver1/Z driver2/Z}] \
  -clock [get_clocks clk] \
  -balance_levels true \
  -target_level 5
```

SEE ALSO

remove_multisource_clock_subtree_options(2)
report_multisource_clock_subtree_options(2)
synthesize_multisource_clock_subtrees(2)

set_multisource_clock_tap_options

Defines constraints for multisource clock tap assignment

SYNTAX

status **set_multisource_clock_tap_options**
[-name *name*]
-clock *clock_name*
-driver_objects *pins_or_ports_or_nets*
-num_taps *value*
[-dont_merge_cells *cell_list*]
[-relax_split_restrictions_for_cells *cell_list*]

Data Types

name string
clock_name collection
pins_or_ports_or_nets collection
value integer
cell_list collection

ARGUMENTS

-name *name*

Specifies a name for the multisource tap option set being constrained. The name can be referenced durin tap assignment. The **-name** option is optional. It allows the user to define multiple option sets per clock. Without this option the constraint set is named as the clock object. Names have to be unique for all defined set of options.

-clock *clock_name*

Specifies a clock which will considered during tap assignment for the specified set of driver objects. This option is required. All sinks reachable form the driver objects for the given clock are reassigned during tap assignment.

If the driver objects do not receive the specified clock, then an error will be issued during **synthesize_multisource_clock_taps**.

-driver_objects *pins_or_ports_or_nets*

This required option specifies driver objects (tap drivers) whose fanouts and sinks can get redistributed among them. If a net name is used, then the driver pin of that net is inferred as the driver object.

For example, if tap driver buffers are driven by the clock mesh directly, then the output pins of the tap drivers could be specified as the driver objects. Alternately, if a top-level mesh is pushed down to the block-level and the mesh straps act as the drivers themselves, then the ports for each mesh strap can be specified as driver objects.

Each driver object should have some common clock propagating through it, because tap assignment is performed for one clock

only.

-num_taps *value*

Specifies the number of tap drivers to be used during tap assignment. For the current version of the tool, the specified value has to equal the number of driver objects. This option has been introduced to support future extensions of the tool where tap drivers will be automatically synthesized.

-dont_merge_cells *cell_list*

This option accepts a collection of cells which should not get merged with other cells by the tap assignment command.

-relax_split_restrictions_for_cells *cell_list*

This option accepts a collection of cells to be considered for splitting even if those cells are set with user_dont_touch/user_size_only. These cells will be split for optimization during tap assignment as required, similar to any other cell without user_dont_touch/user_size_only.

DESCRIPTION

The **set_multisource_clock_tap_options** command is used to apply tap assignment constraints for multisource CTS synthesis using the **synthesize_multisource_clock_taps** command.

Typically, a multisource CTS approach involves some mesh, H-tree, or other regular clock structure that drives a collection of tap driver cells. Sometimes, there may not be explicit tap driver cells, and the mesh straps themselves act as the tap drivers. In either case, the clock trees after the tap drivers are called subtrees, and this command is used to specify the tap drivers and constraints for tap assignment.

Each subtree is driven by a unique driver object, called a tap driver. Multiple subtrees can be grouped together for the purposes of skew balancing, and sink assignment across those subtrees.

This command performs two main functions:

Defines a set of tap drivers for sink assignment
Defines which clock to be considered for sink identification

The **-driver_objects** option defines a set of tap drivers (cell output pins or input ports) which are considered as equivalent for multisource CTS tap assignment. This means that any sink reachable from one of these tap drivers can be reassigned to one of the other driver objects.

By default, a set of tap drivers is named according to the specified clock. In case sets of tap drivers for a particular clock should be kept separate, then the \-name option can be used to distinguish the individual tap driver subsets.

If a particular setting has already been applied for an option set, and then it is applied again for the same option set, then the original setting is overwritten by the new setting.

When a list of cells is specified using the switch relax_split_restrictions_for_cells, tap assignment allows splitting of these cells even if they are set with user_dont_touch/user_size_only.This option is especially helpful with integrated tap assignment in place_opt for cases where certain cells are allowed to split during tap assignment while all other optimizations in place_opt/CTS must honour its user_dont_touch/user_size_only. These cells once split, will not be merged back if synthesize_multisource_clock_taps is called consecutively. Hence, user needs to ensure to use a pre-tap assignment block when exploring tap driver locations using multiple calls to synthesize_multisource_clock_taps.

The **remove_multisource_clock_tap_options** command can be used to clear constraints that have been set with this command.

The **report_multisource_clock_tap_options** command can be used to report constraints set with this command.

The **synthesize_multisource_clock_taps** command is used to synthesize the subtrees, as constrained by this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

This example shows a typical case of how tap driver objects can be assigned to a clock.

```
set_multisource_clock_tap_options -clock [get_clock clk]
 -driver_objects [get_pins {tap_driver1/Z tap_driver2/Z}]
 -num_taps 2
```

This example performs the exact same function as the previous example but makes use of a collection of pins.

```
set tap_drivers [get_pins {tap_driver1/Z tap_driver2/Z}]
set_multisource_clock_tap_options
 -driver_objects $tap_drivers
 -num_taps [sizeof_collection $tap_drivers]
 -clock [get_clocks clk]
```

This example excludes two cells from merging..

```
set_multisource_clock_tap_options
 -driver_objects $tap_drivers
 -num_taps [sizeof_collection $tap_drivers]
 -clock [get_clocks clk]
 -dont_merge_cells [get_cells {u1 u2}]
```

SEE ALSO

remove_multisource_clock_tap_options(2)
report_multisource_clock_tap_options(2)
synthesize_multisource_clock_taps(2)

set_net_estimation_rule

Sets rules for estimating timing on nets.

SYNTAX

status **set_net_estimation_rule**
[-parameter *param_name*]
[-unset]
[-value *value*]
[-horizontal_value *value*]
[-vertical_value *value*]
[-cell *cell*]
rule_name

Data Types

rule_name
param_name string
value string
rule_name string
cell collection

ARGUMENTS

rule_name

Specifies the name of a net estimation rule to set. If you call **set_net_estimation_rule** repeatedly for the same name, you can set or unset various parameter in the rule.

-parameter *param_name*

Specifies a parameter to set or unset. A summary of the available parameters is listed in the "Parameters" section below.

-unset

Removes a manually set value for a rule parameter. This option must be accompanied by *-parameter*.

-value *value*

Sets a value for the given parameter. The value will apply to both horizontal and vertical segments of the nets. Note that different parameters require different kinds of values. Some parameters have string values and others are floating point numbers. This option must be accompanied by *-parameter*.

-horizontal_value *value*

Sets a value for the given parameter. The value will apply only to horizontal segments of the nets. Note that different parameters

require different kinds of values. Some parameters have string values and others are floating point numbers. This option must be accompanied by *-vertical_value* and *-parameter*.

-vertical_value *value*

Sets a value for the given parameter. The value will apply only to vertical segments of the nets. Note that different parameters require different kinds of values. Some parameters have string values and others are floating point numbers. This option must be accompanied by *-horizontal_value* and *-parameter*.

-cell *cell*

Specifies the physical cell for which to create/set the net estimation rule.

By default, the current block is used to set/create the specified net estimation rule.

DESCRIPTION

This command defines parameters to be used when doing timing analysis on nets. The parameters are used, for example, to determine the proper spacing of flip-flops on a pipelined route. Using this command you can specify parameter values for some or all of the parameters described below in the "Parameters" section of this man page.

You can optionally choose to leave some parameters unspecified. In that case, the tool will derive intelligent values for you. For example, if you specify a metal layer to use, the tool can automatically derive the resistance of the wire. The derivation methods are described below.

Whether parameters are manually specified or derived automatically, you can query the parameter values using the **report_net_estimation_rules** or **get_net_estimation_rules** commands. So you can use the **set_net_estimation_rule command** to explore different net configurations. For example, you could explore the effects of changing metal layers on your register spacing.

Note that net estimation rules are only supported for certain tool features. Currently, 2 features support the net estimation rule. Pipeline register planning, using command **create_busplans** supports the net estimation rule and virtual optimization using command **estimate_timing** supports the net estimation rule. In the near future other features will support this. Consult the specific tool feature you are using to determine what exactly is supported.

NET ESTIMATION RULES AND HIERARCHY

Net estimation rules can be created in any physical block at any level of the hierarchy and multiple blocks can have different rules with the same name. However, rules are often stored/retrieved by name, and doing this can cause confusion as to which rule is being used by a given command. Generally, the "current" (top) block's list of rules is searched by commands such as *estimate_timing*. This means that if you define rule R1 in "top" and rule R1 in block "B1", if you make block B1 the current block, commands will use B1's R1. When you go back to top, top's R1 is used. To avoid confusion, use globally unique rule names and define them at the top - they will be propagated to child blocks during distributed commands as necessary. Rules in child blocks that have the same name as rules in top may also be overwritten during distributed commands.

Parameters

layer

Specify the preferred metal layer on which the nets should be routed. The value should be the name of a routing layer in your technology. You can set a different value for horizontal and vertical. If you do not set a value, the middle layer of the current min and max routing layers will be used.

utilization

Specify the routing utilization that should be assumed for the metal layer you have chosen. The value should be a floating point

number between 0 and 1.0, where 1.0 is 100% utilization. You can set a different value for horizontal and vertical. If you do not set a value, 0.7 will be assumed.

delay_derate

Specify a derating factor for ns_per_mm. You must not specify a separate value for horizontal and vertical. If you do not set a value, default is 1.

sdc_stage_delay

Specify the clock period minus clock uncertainty. You must not specify a separate value for horizontal and vertical. If you do not set a value it will be derived based on the clock parameter.

ndr

Specify an NDR routing rule to be used by your nets wires. The value should be the name of an existing NDR rule. You must not specify a separate value for horizontal and vertical. If you do not set a value, the default routing rule will be used.

corner

Specify the optimization corner to be used when analyzing nets. You must not specify a separate value for horizontal and vertical. If you do not set a value, the slowest active corner will be used.

ff_per_mm

Specify a value for the capacitance of your nets wires (in ff per mm). You can set a different value for horizontal and vertical. If you do not set a value, the value will be derived based on the metal layer, ndr rule, corner, xtalk_fraction, and utilization.

ohms_per_mm

Specify a value to set the resistance of your nets wires (in ohms per mm). You can set a different value for horizontal and vertical. If you do not set a value, the value will be derived based on the metal layer, ndr rule, corner, and utilization.

buffer

Specify the buffer or inverter that you would like along your nets. The value should be the name of a buffer or inverter in your library. You must not specify a separate value for horizontal and vertical. If you do not set a value, a good buffer will automatically be chosen based on your corner, ff_per_mm, ohms_per_mm.

buffer_spacing

Specify the amount of distance that you want between the buffers on your nets (in microns). You can specify a separate value for horizontal and vertical. If you do not set a value, an optimal value will be calculated based on your corner, ff_per_mm, ohms_per_mm and buffer.

buffer_pattern

Specify the buffer pattern name (string) uesd to insert buffers into the corresponding topology edge. Separate horizontal and vertical buffer patterns are supported, or they can both be the same.

ns_per_mm

Specify the speed of propagation along a buffered net (in ns per mm). This will include both buffer delay time and wire delay time. You can specify a separate value for horizontal and vertical. If you do not set a value, the value will be calculated based on your corner, ff_per_mm, ohms_per_mm, buffer, delay_derate, and buffer_spacing.

clock

Specify the name of the clock used by registers along your path. You must not specify a separate value for horizontal and vertical. If you do not set a value, the value will be net specific and will be taken from the clock on the flip-flop that drives the net (if available).

stage_margin

Specify extra timing margin in pipeline stages along a path (in ns). You must not specify a separate value for horizontal and vertical. If you do not set a value, zero will be assumed.

register

Specify the name of the flip-flop to use for pipeline stages along your path. You must not specify a separate value for horizontal and vertical. If you do not set a value, an ideal register (no delay) will be assumed for timing calculations.

register_spacing

Specify the maximum spacing of flops along a pipeline path. You can set a different value for horizontal and vertical. If you do not set a value, the value will be derived based on the corner, register, clock, stage_margin, and ns_per_mm.

register_pattern

Specify the register pattern name (string) uesd to insert registers into the corresponding topology edge. Separate horizontal and vertical register patterns are supported, or they can be the same.

launch_block_budget

Specify the delay used inside of your driving block before reaching the block's output pin. You must not set a different value for horizontal and vertical. If you do not set a value, the value will zero.

capture_block_budget

Specify the delay used inside of your load block after reaching the block's input pin. You must not set a different value for horizontal and vertical. If you do not set a value, the value will zero.

EXAMPLES

The following example sets the corner parameter for the net estimation rule named rule1.

```
prompt> set_net_estimation_rule rule1 -parameter corner -value WCCOM
```

The following example sets the layer parameter for the net estimation rule named rule1.

```
prompt> set_net_estimation_rule rule1 -parameter layer \
  -horizontal_value M4 -vertical_value M5
```

The following example sets the buffer parameter for the net estimation rule named rule1.

```
prompt> set buffer BUFX8
prompt> set_net_estimation_rule rule1 -parameter buffer -value $buffer
```

The following example sets the buffer and layer parametesr for the net estimation rule named rule1, and then set rule1 as the *estimate_timing_net_rule* attribute for net n101 so *estimate_timing* would use the *ns_per_mm* value of rule1 to estimate the delay on net n101.

```
prompt> set buffer BUFX8
prompt> set_net_estimation_rule rule1 -parameter buffer -value $buffer
prompt> set_net_estimation_rule rule1 -parameter layer \
  -horizontal_value M4 -vertical_value M5
prompt> set_attribute -name estimate_timing_net_rule -value rule1 [get_nets n101]
prompt> estimate_timing
```

SEE ALSO

get_net_estimation_rules(2)
remove_net_estimation_rules(2)
report_net_estimation_rules(2)

set_net_type

Set the net type of a net.

SYNTAX

status **set_net_type**
-net *net*
-type *net_type*

Data Types

net collection
net_type string

ARGUMENTS

-net *net*

Specifies the net whose type will be set. This is a mandatory option.

-type *net_type*

Specifies the type of the net. Allowed values are *analog_ground*, *analog_power*, *analog_signal*, *clock*, *deep_nwell*, *deep_pwell*, *ground*, *nwell*, *power*, *pwell*, *reset*, *scan*, *signal*, *tie_high*, *tie_low and unset. This is a mandatory option.*

DESCRIPTION

The **set_net_type** command sets the net type of the given net by changing the net type of all the segments of that net.

EXAMPLES

The following example sets the net type of the net u1/i1 to power.

prompt> **set_net_type -net u1/i1 -type power**

SEE ALSO

set_attribute(2)

set_net_weight_effort

Sets the net weight effort for coarse placement.

SYNTAX

status **set_net_weight_effort**
-nets *collection_of_nets*
-weight_effort low | medium | high | ultra | none
-virtual_flat
-verbose

Data Types

collection_of_nets collection

ARGUMENTS

-nets *collection_of_nets*

Specifies the collection of nets for setting the weight effort.

-weight_effort low | **medium** | **high** | **ultra** | **none**

Specifies the weight effort to be used in coarse placement. The default for a net is low. If set to "none", the placer will ignore the net.

-virtual_flat

Propagate the net weight on the input top-level net to associated segments of the same flat net in all child blocks.

-verbose

If true, output further information about nets (such as which dangling nets are being skipped).

DESCRIPTION

The **set_net_weight_effort** command sets the net weight effort on nets for coarse placement. Note that effort might not always be strictly followed for high-fanout nets or for situations with other competing priorities, such as density, exist. Use the **report_net_weight_effort** command to list weight efforts for nets, and the **remove_net_weight_effort** command to remove them.

EXAMPLES

The following example sets the net weight effort for the net n123 to medium.

```
prompt> set_net_weight_effort -nets [get_nets n123] -weight_effort medium
```

SEE ALSO

remove_net_weight_effort(2)
report_net_weight_effort(2)

set_noise_margin

Specifies the noise margin for a library pin, port, or pin.

SYNTAX

status **set_noise_margin**
[-above]
[-below]
[-low]
[-high]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
margin_value
object_list

Data Types

mode_list list
corner_list list
scenario_list list
margin_value float
object_list list

ARGUMENTS

-above

Specifies the noise margin for above ground or power rail noise analysis region.

-below

Specifies the noise margin for below ground or power rail noise analysis region.

-low

Specifies the noise margin for ground rail noise.

-high

Specifies the noise margin for power rail noise.

-modes *mode_list*

Specifies the scenarios on which to set the noise margin value. If this option is given, all scenarios of the given modes are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose

mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-corners ***corner_list***

Specifies the scenarios on which to set the noise margin value. If this option is given, all scenarios of the given corners and the current mode is used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-scenarios ***scenario_list***

Specifies the scenarios on which to set the noise margin value. The **-modes** or **-corners** option must not be given with this option.

margin_value

Specifies a margin value. The value is the input height in voltage units.

object_list

Specifies a list of library pins, ports, or pins.

DESCRIPTION

Each library cell input can tolerate a certain amount of noise without causing a failure at the cell output. This characteristic is called noise immunity. This command specifies noise immunity at library pins to determine whether noise failures occur as well as the amount of noise slack.

Noise immunity in the library can be specified as noise immunity curves, polynomials, or tables or in terms of noise margins.

This command specifies a noise margin for a library pin, design port, or pin. It can be used in the absence of library-specified noise immunity characteristics, or to override the library-specified characteristics by replacing them with noise margins. For a library pin, this command is set for current corner when without **-corners** option, and only supports **-corners** option to specify corners. For a design port or pin, this command is for current scenario when without **-scenarios**, **-corners** and **-modes** options, or using the three options to specify scenarios.

Noise margins consider only the bump heights at the cell inputs. Using height-only noise margins are faster and more conservative than the other methods.

For high, narrow noise bumps, using height-only noise margins is pessimistic because it treats some bumps as noise failures that would otherwise pass with the immunity curve model. However, for wide noise bumps, using noise margins gives the same results as using noise immunity curves.

Noise immunity characteristics can vary for different noise bump types, so there can be four different noise margins associated with each input: below low, above low, below high, and above high. Specify all values as positive numbers for all four types of noise bumps.

EXAMPLES

This example specifies a noise margin of 4 for above-the-ground rail noise for the A pin of the IV library cell in the lsi_10k library:

prompt> **set_noise_margin -above -low 4 lsi_10k/IV/A**

SEE ALSO

remove_noise_margin(2)

set_object_layer

Sets the layer(s) of different objects in the current design.

SYNTAX

collection **set_object_layer**
[-force]
-of_objects collection
-layers collection

ARGUMENTS

-force

Ignore locked status of the objects. If this option is not provided and any of the affected objects has locked status, command will fail generating tcl error.

-of_objects *collection*

Specifies the collection of objects whose layer needs to be modified. This can be a list of strings or output of *get_* commands.

-layers *collection*

Specifies the collection of layers. This can be a list of strings or output of *get_layers* command.

DESCRIPTION

The **set_object_layer** command enables you to modify the layers of some of the objects. The type of objects whose layer can be modified are shapes, terminals, pin_blockages, pin_guides, routing_blockages and routing_guides.

Changing layer of shapes and terminals creates a new shape object and deletion of existing shape. For terminals, the new shape object is attached to the terminal. For shapes, terminals and routing blockages only the first layer is picked from the collection of layers specified.

EXAMPLES

The following example sets layer of shape RECT_32 to METAL2. The RECT_32 shape will no longer exist. A new shape will be created and returned by the command.

prompt> **set_object_layer -of_objects [get_shapes RECT_32] \ -layers METAL2**

The following example sets layer of routing blockage RB_11 to METAL2 and layers of pin guide PG_10 to METAL2 and METAL3.

prompt> **set_object_layer -of_objects "RB_11 PG_10" \ -layers "METAL2 METAL3"**

set_object_shape

Sets the shape of a set of objects to one of the standard rectilinear shapes.

SYNTAX

status **set_object_shape**
-shape *shape_type*
-lengths *list_of_lengths*
[-utilization *utilization*]
[-area *area*]
[-keep_area]
[-rotate 0 | 90 | 180 | 270 | CW90 | CW180 | CW270 | CCW90 | CCW180 | CCW270 | FLIPX | FLIPY]
[*object_list*]

Data Types

shape_type string
list_of_lengths list
utilization real
area real
object_list collection or selection

ARGUMENTS

object_list

Collection or selection set containing objects to set the shape. If this option is not specified, the global selection set is used.

-shape *shape_type*

This is either a rectangular (**rect**), L (**l**), T (**t**), U (**u**), or cross (**cross**) shape.

Only objects that can be resized can be set to a **rect** shape.

Only objects that can be rectilinear can be set to a **l**, **t**, **u**, or **cross** shape.

This is a required option.

-lengths *lengths*

Specifies the absolute lengths in microns or length ratios of the sides of the shape.

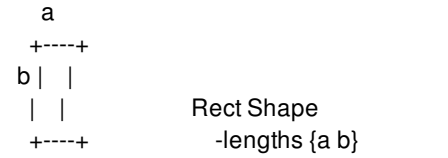
Absolute lengths are assumed unless you use one of the following options: **-utilization**, **-area**, or **-keep_area**.

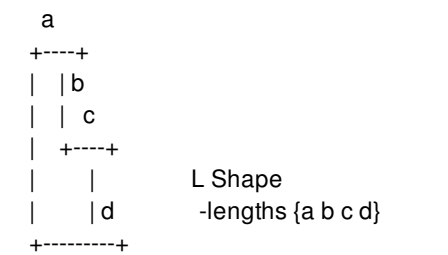
If you specify length ratios, the resultant shape has sides with the corresponding ratios. For example, if the ratios are all one, all lengths are the same. If one ratio is two and the others are one, that side is twice as long as the others.

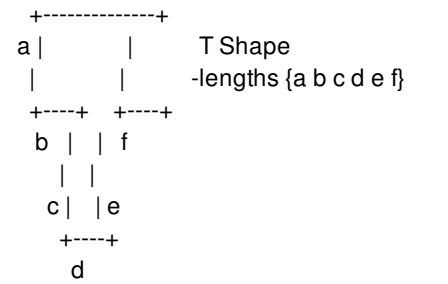
Each shape requires a different number of values as described below:

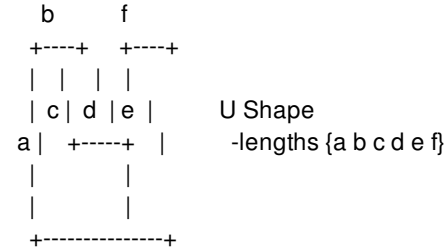
- rect - requires 2 values
- l - requires 4 values
- t - requires 6 values
- u - requires 6 values
- cross - requires 6 values

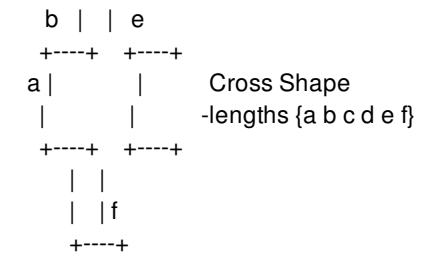
See the following diagrams for a cross reference between length parameter positions in the list and the corresponding edge on each shape, where a is first value, b is second value, and so on:











This is a required option.

-utilization ***utilization***

Specifies that the new shape should have the specified utilization.

Utilization is defined as the physical_area divided by the placement area and is most commonly used for placement objects and utilization where the enclosed area can be larger than the associated child physical objects.

Utilization must be positive and nonzero.

Note: This implies that lengths are ratios and not absolute values.

-area ***area***

Specifies that the new shape should have the specified area.

Note: This implies that lengths are ratios and not absolute values.

The default is 1.0.

-keep_area

Specifies that the new shape should have the same area as the current object.

Note: This implies that lengths are ratios and not absolute values.

-rotate 0 | **90** | **180** | **270** | **CW90** | **CW180** | **CW270** | **CCW90** | **CCW180** | **CCW270** | **FLIPX** | **FLIPY**

Specifies that the base shape is geometrically rotated from the default orientation (see the diagram above) before being applied to the object.

Note: This option cannot be used with the **-shape rect** option.

The default is 0.

DESCRIPTION

This command sets the shape of the boundary of one or more objects.

EXAMPLES

The following example sets the boundary of the selected objects to an L shape with all sides of equal length and a utilization of 0.7.

```
prompt> set_object_shape -shape l -lengths {1 1 1 1} -utilization 0.7
```

The following example uses an alternate syntax to specify an L shape for the current collection of selected objects.

```
prompt> set_object_shape [get_selection] -shape l -lengths {1 1 1 1} -utilization 0.7
```

SEE ALSO

get_selection(2)
change_selection(2)

set_operating_conditions

Specifies the operating conditions that are used to establish the process number, voltage, and temperature for the current corner of the current design.

SYNTAX

status **set_operating_conditions**
[-analysis_type single | bc_wc | on_chip_variation]
[-library *lib*]
[*condition*]
[-min *min_condition*]
[-max *max_condition*]
[-min_library *min_lib*]
[-max_library *max_lib*]
[-object_list *objects*]

Data Types

lib	collection
condition	string
min_condition	string
max_condition	string
min_lib	collection
max_lib	collection
objects	list

ARGUMENTS

-analysis_type single | bc_wc | on_chip_variation

This option is ignored, as the tool always uses the **on_chip_variation** analysis type.

-library *lib*

Specifies the library that contains the definitions of the operating conditions to be used. This is either a library name or a collection object. If the name refers to a logic library file, the command uses the reference library that was built from that logic library.

condition

Specifies the name of the operating condition use for both minimum and maximum delays and hold violations.

-min *min_condition*

Specifies the operating condition used for checking minimum delays and hold violations.

If you use this option, you must also use the **-max** option.

-max ***max_condition***

Specifies the operating condition used for checking maximum delays and setup violations.

If you use this option, you must also use the **-min** option.

-min_library ***min_lib***

Specifies the library that contains the minimum operating condition. This is either a library name or a collection object. If the name refers to a logic library, the command uses the reference library that was built from that logic library.

If you use this option, you must also use the **-min** option.

-max_library ***max_lib***

Specifies the library that contains the maximum operating condition. This is either a library name or a collection object. If the name refers to a logic library, the command uses the reference library that was built from that logic library.

If you use this option, you must also use the **-max** option.

-object_list ***objects***

Specifies the cells or ports on which to set the operating conditions.You can specify both leaf cells and hierarchical blocks.

If you do not use this option, operating conditions are set on the design.

You cannot use this option with the **-analysis_type** option.

DESCRIPTION

Specifies the operating conditions that are used to establish the process number, voltage, and temperature for the current corner of the current design. The tool treats an operating condition as an alias for a set of values for process number, voltage, and temperature.

The specified operating conditions must be defined in one of the reference libraries specified in the **-library** option, the **-min_library** option (for the minimum operating condition only), or the **-max_library** option (for the maximum operating condition only).

Operating conditions set by using the **-object_list** option override the operating conditions set on the design or on higher levels of hierarchy.

You can also specify the desired PVT parameters with the **set_process_label**, **set_process_number**, **set_voltage**, and **set_temperature** commands. You should avoid using the **set_operating_conditions** command for new designs, and use the commands mentioned above instead. Reference libraries can contain multiple operating conditions with the same name, which were obtained from the various logic libraries used to build them.

If you do not specify any operating conditions or PVT parameters for a design, the tool uses the default PVT conditions of the library to which the cell is linked.

To view the operating conditions and other PVT parameters being used for any particular corner, use the **report_corner** command. To view the operating conditions defined in a particular library, use the **report_lib** command. To view the PVT parameters appplicable to any particular cell or block, use the **report_cell** or **report_pvt** commands.

To remove the operating condition settings from a corner, use the **remove_sdc** or **reset_design** command.

Multicorner-Multimode Support

This command works only on the current corner.

EXAMPLE

SEE ALSO

set_process_label(2)
set_process_number(2)
set_voltage(2)
set_temperature(2)
report_cell(2)
report_corner(2)
report_pvt(2)
report_lib(2)
reset_design(2)
remove_sdc(2)

set_optimize_registers

Adds (or removes) retiming on specific modules during the **compile** command

SYNTAX

int **set_optimize_registers**
[-modules *module_list*]
[-print_critical_loop]
[-justification_effort low | medium | high]
[true | false]

Data Types

module_list collection

ARGUMENTS

-modules *module_list*

Specifies a collection of modules that should (or should not) be retimed during the **compile** command. The default is the module of the current design.

-print_critical_loop

Indicates that the critical loop of the design, as seen during retiming, is to be displayed. The critical loop is defined as the sequence of directly-connected combinational and sequential cells whose total combinational delay divided by the number of registers in the loop has a higher value than any other loop in the design. The critical loop limits the minimum clock period that can be achieved by retiming. Use this option to help troubleshoot problem areas of the design if the intended clock period cannot be achieved with the given number of sequential cells in the design. If you are pipelining a data path, you might need to add pipeline stages in the HDL code.

-justification_effort low | medium | high

Specifies the justification effort level that is to be used during backward justification of registers. You can specify a value of low, medium, or high. Specifying low ensures that justification terminates very quickly, but the QoR may be bad. Specifying medium provides good QoR with considerable runtime. Specifying high provides the best QoR without any regard for runtime. The default value for this option is **medium**.

true | false

For a given set of modules, the value **true** adds retiming to the **compile** command's optimizations, and the value **false** removes retiming from the **compile** command's optimizations. The default is *true*.

DESCRIPTION

The **set_optimize_registers** command adds retiming to the **compile** command's optimizations for a specified set of modules.

Retiming is a sequential optimization that moves registers across combinational logic to meet a target clock period or, if that is not possible, to minimize negative slack. Retiming raises the level of design abstraction: RTL modules targeted for retiming need not be pipelined explicitly but instead can be described as combinational datapaths with banks of registers at the I/Os. Such RTL-style increases the area reduction opportunities for many optimizations, especially datapath extraction. Since retiming occurs at the gate-level instead the RTL operator-level, it can often find solutions that are difficult or impossible to express in RTL. Retiming is ideally suited for balancing multiple pipeline stages within combinational logic and hence is primarily applicable to datapath blocks.

When retiming is invoked during the **compile** command, the target clock period is extracted from the timing requirements of the module being retimed. Within a targeted module, retiming may create new and/or remove existing registers.

To remove retiming from the **compile** command's optimizations, execute the **set_optimize_registers** command with the last parameter set to **false**.

EXAMPLES

In the following example, retiming is enabled on the module **TEST**.

prompt> **set_optimize_registers -modules [get_modules TEST]**

In the following example, retiming is enabled on the current design and disabled on the module **OLD**.

prompt> **set_optimize_registers**

prompt> **set_optimize_registers -modules [get_modules OLD] false**

SEE ALSO

compile(2)
current_design(2)
get_modules(2)
set_dont_retime(2)

set_output_delay

Sets output path delay values for the current design.

SYNTAX

string **set_output_delay**
[-clock *clock_name*]
[-reference_pin *pin_port_name*]
[-clock_fall]
[-level_sensitive]
[-rise]
[-fall]
[-max]
[-min]
[-add_delay]
[-network_latency_included]
[-source_latency_included]
[-group_path *group_name*]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
delay_value
port_pin_list

Data Types

clock_name list
pin_port_name list
group_name string
mode_list list
corner_list list
scenario_list list
delay_value float
port_pin_list list

ARGUMENTS

-clock *clock_name*

Specifies the name of a clock to which to relate the specified delay. If you specify **-clock_fall**, you must also specify **-clock**.

-reference_pin *pin_port_name*

Specifies the clock pin or port to which to relate the specified delay. If you use this option, and if propagated clocking is being used, the delay value is related to the arrival time at the specified reference pin, which is clock source latency plus its network latency from the clock source to this reference pin. The options **-network_latency_included** and **-source_latency_included** cannot be used together with the **-reference_pin** option. For ideal clock network, the delay is the sum of source latency and network latency applied.

The pin specified with the **-reference_pin** option should be a leaf pin or port in a clock network, in the direct or transitive fanout of a clock source specified with the **-clock** option. If multiple clocks reach the port or pin where you are setting the input delay, and if the **-clock** option is not used, the command considers all of the clocks.

-clock_fall

Specifies that the delay is relative to the falling edge of the clock. If you specify **-clock_fall** with **-reference_pin**, the delay is relative to the falling transition of the reference pin. If you specify **-clock**, the default is the rising edge or the rising transition of the reference pin. If you specify **-clock_fall**, you must also specify **-clock**.

-level_sensitive

Specifies that the destination of the delay is a level-sensitive latch. The tool derives the setup and hold relationships for paths to this port as if it were a level-sensitive latch. If you do not use **-level_sensitive**, the output delay is treated as if it were a path to a flip-flop.

-rise

Specifies that *delay_value* refers to a rising transition on specified ports of the current design. If you do not specify **-rise** or **-fall**, rising and falling delays are assumed to be equal.

-fall

Specifies that *delay_value* refers to a falling transition on the specified ports of the current design. If you do not specify **-rise** or **-fall**, rising and falling delays are assumed to be equal.

-max

Specifies that *delay_value* refers to the longest path. If you do not specify **-max** or **-min**, maximum and minimum output delays are assumed to be equal.

-min

Indicates that *delay_value* refers to the shortest path. If you do not specify **-max** or **-min**, maximum and minimum output delays are assumed to be equal.

-add_delay

Specifies that delay information is to be added to the existing output delay, instead of overwriting the output delay. Using **-add_delay**, you can capture information about multiple paths leading from an output port that are relative to different clocks or clock edges. For example, **set_output_delay 5.0 -max -rise -clock phi1 {OUT1}** removes all other maximum rise output delays from **OUT1**, because **-add_delay** is not specified. Other output delays with a different clock or with **-clock_fall** are removed.

In another example, **-add_delay** is specified: **set_output_delay 5.0 -max -rise -clock phi1 -add_delay {Z}**. If there is an output maximum rise delay for Z relative to the clock phi1 rising edge, the larger value is used. The smaller value does not result in critical timing for maximum delay. For minimum delay, the smaller value is used. If there is a maximum rise output delay relative to a different clock or different edge of the same clock, it remains with the new delay.

-network_latency_included

Specifies whether the clock network latency should not be added to the output delay value. If this option is not specified, the clock network latency of the related clock will be added to the output delay value. It has no effect if the clock is propagated or the output delay is not specified with respect to any clock.

-source_latency_included

Specifies whether the clock source latency should not be added to the output delay value. If this option is not specified, the clock source latency of the related clock will be added to the output delay value. It has no effect if the output delay is not specified with respect to any clock.

-group_path ***group_name***

Specifies the name of a group into which paths ending at the specified ports or pins are to be added. If the group does not already exist, one is created. This is equivalent to specifying the separate command **group_path -name group_name -to port_pin_list** in addition to **set_output_delay**. If you do not specify **-group_path**, the existing path grouping does not change.

-modes ***mode_list***

Applies the delay value to the specified scenarios. If this option is given, all scenarios of the given modes will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by a **-corners** will be used.

-corners ***corner_list***

Applies the delay value to the specified scenarios. If this option is given, all scenarios of the given corners and the current mode will be used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are given, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** will be used.

-scenarios ***scenario_list***

Specifies the scenarios that the delay value will be applied to. The **-modes** or **-corners** option must not be specified with this option.

delay_value

Specifies the path delay in units consistent with the technology library used during analysis. The *delay_value* represents the amount of time before a clock edge that the signal is required. For maximum output delay, this represents a combinational path delay to a register plus the library setup time of that register. For minimum output delay, this value is usually the shortest path delay to a register minus the library hold time.

port_pin_list

Specifies a list of output port or internal pin names in the current design to which *delay_value* is assigned. If more than one object is specified, the objects are enclosed in braces ({}).

DESCRIPTION

This command sets output path delay values for the current design. The input and output delays characterize the operating environment of the current design when used with **set_load** and **set_driving_cell**.

The **set_output_delay** command sets output path delays on output ports relative to a clock edge. Output ports have no output delay unless specified. For inout (bidirectional) ports, you can specify the path delays for both input and output modes.

To describe a path delay to a level-sensitive latch, use the **-level_sensitive** option. If the latch is positive-enabled, set the output delay relative to the rising clock edge. If the latch is negative-enabled, set the output delay relative to the falling clock edge. If time is borrowed at that latch, subtract that time from the path delay to the latch when determining output delay.

The tool adds input delay to path delay for paths starting at primary inputs and to output delay for paths ending at primary outputs.

The **-group_path** option modifies the path grouping. Path grouping affects the maximum delay cost. The worst violator within each group adds to the cost.

If a reference pin is not reachable by any given clock, zero arrival time is assumed. If no clock is specified with a reference pin and

this reference pin is not reachable by any clock, an unconstrained path is reported. This behavior is changed after X0506. The new behavior is these invalid constraints will be ignored and path will be constrained by whatever it should be as if no such constraints are defined at all. The **check_timing** command generates a warning if the reference pin is not reachable by any active clock, or if multiple clocks reach the reference pin and no clock is specified in the command.

Output delays are managed on a per-scenario basis. For designs with multiple scenarios, you can specify different delays for different scenarios by using **-modes**, **-corners**, and **-scenarios** options.

By default, if none of these options are specified, the given delay is applied to the current scenario. It is an error if there is no current scenario. If the **-scenarios** option is given, the delay will be applied to all of the specified scenarios. If the **-modes** option is given, the delay will be applied to all scenarios of the specified modes. It is an error if none of the specified modes have any scenarios. If the **-corners** option is given, the delay will be applied to all scenarios of the specified corners and the current mode. It is an error if none of the specified corners have any scenarios with the current mode. If both the **-modes** and **-corners** options are given, the delay will be applied to all scenarios of the specified corners and the specified modes. It is an error if there are no such scenarios. It is an error to specify **-scenarios** with **-modes** or **-corners**. To list output delays associated with ports, use **report_ports**.

To remove output delay values, use **remove_output_delay** or **reset_design**. To modify paths grouped with the **-group_path** option, use the **group_path** command to place the paths in another group or the default group.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example sets an output delay of 1.7 relative to the rising edge of CLK1 for all output ports in the design.

```
prompt> set_output_delay 1.7 -clock CLK1 [ all_outputs ]
```

The following example sets the input and output delays for the bidirectional port INOUT1. The input signal arrives at INOUT1 2.5 units after the falling edge of CLK1. The output signal is required at INOUT1 at 1.4 units before the rising edge of CLK2.

```
prompt> set_input_delay 2.5 -clock CLK1 -clock_fall { INOUT1 }
prompt> set_output_delay 1.4 -clock CLK2 { INOUT1 }
```

The following example models the situation where there are three paths from output port OUT1. One of the paths is relative to the rising edge of CLK1. Another path is relative to the falling edge of CLK1. The third path is relative to the falling edge of CLK2. The **-add_delay** option indicates that new output delay information does not cause the removal of old information.

```
prompt> set_output_delay 2.2 -max clock CLK1 -add_delay { OUT1 }

prompt> set_output_delay 1.7 -max clock CLK1 -clock_fall -add_delay { OUT1 }

prompt> set_output_delay 4.3 -max clock CLK2 -clock_fall -add_delay { OUT1 }
```

In the following example, two different maximum delays and two minimum delays for port Z are specified using **-add_delay**. Because the information is relative to the same clock and clock edge, only the largest of the maximum values and the smallest of the minimum values are maintained (in this case, 5.0 and 1.1). If **-add_delay** is not used, the new information overwrites the old information.

```
prompt> set_output_delay 3.4 -max -clock CLK1 -add_delay { Z }

prompt> set_output_delay 5.0 -max -clock CLK1 -add_delay { Z }
```

prompt> **set_output_delay 1.1 -min -clock CLK1 -add_delay { Z }**

prompt> **set_output_delay 1.3 -min -clock CLK1 -add_delay { Z }**

The following example uses the **-group_path** option to add ports into a named group. Note that without this option, paths to these ports are included in the CLK group.

prompt> **set_output_delay 4.5 -max -clock CLK -group_path busA {busA[*]}**

SEE ALSO

all_outputs(2)
create_clock(2)
current_design(2)
group_path(2)
remove_output_delay(2)
report_design(2)
report_ports(2)
reset_design(2)
set_driving_cell(2)
set_load(2)
set_max_delay(2)
set_output_delay(2)

set_parasitic_parameters

Sets parasitic parameters such as parasitic spec (TLU+) and temperature.

SYNTAX

string **set_parasitic_parameters**
[-corners *corners*]
[-early_spec *early_spec*]
[-early_temperature *early_temperature*]
[-late_spec *late_spec*]
[-late_temperature *late_temperature*]
[-library *library_name*]

Data Types

corners list
early_spec string
early_temperature float
late_spec string
late_temperature float
library_name string

ARGUMENTS

-corners

Sets the parasitic parameter to the constraint corner. If this option is not specified, current conrer will be applied.

-early_spec *early_spec*

Specifies the early parasitic tech specification name.

-early_temperature *early_temperature*

Specifies the early temperature value.

-late_spec *late_spec*

Specifies the late parasitic tech specification name.

-late_temperature *late_temperature*

Specifies the late temperature value.

-library *library_name*

Specifies the library name that contains the specified parasitic specification.

DESCRIPTION

Sets the parasitic parameters required for RC extraction. The command associates the parasitic parameters with the default constraint corner or a list of constraint corners.

Multicorner-Multimode Support

EXAMPLES

The following example sets parasitic parameters to the current constraint corner.

```
prompt> set_parasitic_parameters -early_spec file.tlup -early_temperature 25
1
```

SEE ALSO

all_corners(2)
create_corner(2)
create_mode(2)
current_corner(2)
get_corners(2)
remove_corner(2)
report_extraction_options(2)
report_parasitic_parameters(2)
set_extraction_options(2)

set_parasitics_derate

Sets parasitic derating factors that influence extraction for DR/GR/CTO/RDE. Basically, most options are in aglignment with PrimeTime behavior.

SYNTAX

string **set_parasitics_derate**
[-corners *corners*]
[-ground_capacitance_factor *gcap_scale*]
[-coupling_capacitance_factor *ccap_scale*]
[-resistnace_factor *res_scale*]
[-cap_scale_by_layer *layer_cap_scale*]
[-ccap_scale_by_layer *layer_coupling_cap_scale*]
[-res_scale_by_layer *layer_res_scale*]
[-min | -max]
[-data | -clock]

Data Types

corners	list
gcap_scale	float
ccap_scale	float
res_scale	float
layer_cap_scale	string
layer_coupling_cap_scale	string
layer_res_scale	string

ARGUMENTS

-corners *corners*

Specifies the list of corners for which to apply the extraction options. Derating factors will be applied to default corner without -corners.

-ground_capacitance_factor *gcap_scale*

Specifies the ground capacitance derating factor. In pre-route stage, it works as total capacitance derating factor.

-coupling_capacitance_factor *ccap_scale*

Specifies the coupling capacitance derating factor. It affects coupling capacitance of total capacitance without SI impact. It doesn't change coupling capacitance. It is in alignment with PrimeTime behavior.

-resistance_factor *res_scale*

Specifies the resistance derating factor.

-cap_scale_by_layer ***layer_cap_scale***

Specifies layer based capacitance derating factors.

-caap_scale_by_layer ***layer_ccap_scale***

Specifies layer based resistance derating factors.

-res_scale_by_layer ***layer_res_scale***

Specifies layer based resistance derating factors.

-min

Specifies derating factor for min analysis.

-max

Specifies derating factor for max analysis.

-data

Specifies derating factor for signal nets.

-clock

Specifies derating factor for clock nets.

DESCRIPTION

This command sets derating factors used during RC extraction. The derating options will be associated with default constraint corner, or each list of constraint corners.

The default derating factor value is 1.0, which means no derating. This command supports 0 and positive derating factor value, and checks and warns when abnormal negative derating factors are specified.

Multicorner-Multimode Support

EXAMPLES

In the following example, extraction options are applied to the current constraint corner.

```
prompt> set_parasitics_derate -max -ground_capacitance_factor 1.1 -coupling_capacitance_factor 1.05 -resistance_factor 1.01
1
prompt> report_parasitics_derate

Global factors:
---------------

Corner: default
* max
```

```
  Cap factor      : 1.1
  Res factor      : 1.01
  Coupling factor  : 1.05
1
prompt> set_parasitics_derate -cap_scale_by_layer {{M1 0.99} {M3 0.98} {M5 1.01}} -min
1
prompt> set_parasitics_derate -cap_scale_by_layer {{M5 1.01}} -max
1
prompt> report_parasitics_derate

Global factors:
---------------

Corner: default
* max
  Cap factor      : 1.1
  Res factor      : 1.01
  Coupling factor  : 1.05

  Layer based scale
  -----------------
    Cap scale
      M3        : 0.98
      M5        : 1.01
      M1        : 0.99
* min

  Layer based scale
  -----------------
    Cap scale
      M5        : 1.01
      M1        : 0.99
      M3        : 0.98
1
prompt> set_parasitics_derate -max -ground_capacitance_factor 1.3 -coupling_capacitance_factor 1.1 -resistance_factor 1.02 -clo
1
prompt> report_parasitics_derate

Global factors:
---------------

Corner: default
* max
  Cap factor(Data,Clock)         : {1.1, 1.3}
  Res factor(Data,Clock)         : {1.01, 1.02}
  Coupling factor(Data,Clock)    : {1.05, 1.1}

  Layer based scale
  -----------------
    Cap scale
      M3        : 0.98
      M5        : 1.01
      M1        : 0.99
* min

  Layer based scale
```

```
    -----------------
      Cap scale
        M5        : 1.01
        M1        : 0.99
        M3        : 0.98
1
```

SEE ALSO

all_corners(2)
create_corner(2)
create_mode(2)
current_corner(2)
get_corners(2)
remove_corners(2)
report_parasitics_derate(2)
reset_parasitics_derate(2)
report_parasitic_parameters(2)
set_parasitic_parameters(2)
get_user_units(2)

set_parasitics_parameters

Sets parasitic parameters such as parasitic spec (TLU+) and temperature.

SYNTAX

string **set_parasitic_parameters**
[-corners *corners*]
[-early_spec *early_spec*]
[-early_temperature *early_temperature*]
[-late_spec *late_spec*]
[-late_temperature *late_temperature*]
[-library *library_name*]

Data Types

corners list
early_spec string
early_temperature float
late_spec string
late_temperature float
library_name string

ARGUMENTS

-corners

Sets the parasitic parameter to the constraint corner. If this option is not specified, current conrer will be applied.

-early_spec *early_spec*

Specifies the early parasitic tech specification name.

-early_temperature *early_temperature*

Specifies the early temperature value.

-late_spec *late_spec*

Specifies the late parasitic tech specification name.

-late_temperature *late_temperature*

Specifies the late temperature value.

-library *library_name*

Specifies the library name that contains the specified parasitic specification.

DESCRIPTION

Sets the parasitic parameters required for RC extraction. The command associates the parasitic parameters with the default constraint corner or a list of constraint corners.

EXAMPLES

The following example sets parasitic parameters to the current constraint corner.

```
prompt> set_parasitic_parameters -early_spec file.tlup -early_temperature 25
1
```

SEE ALSO

all_corners(2)
create_corner(2)
create_mode(2)
current_corner(2)
get_corners(2)
remove_corner(2)
report_extraction_options(2)
report_parasitic_parameters(2)
set_extraction_options(2)

set_partial_on_translation

Defines the translation of PARTIAL_ON to FULL_ON or OFF for purposes of evaluating the power state of supply sets and power domains.

SYNTAX

status **set_partial_on_translation**
[*default_translation*]
[-full_on_tools {*tools*}]
[-off_tools {*tools*}]

Data Types

default_translation string
tools list

ARGUMENTS

default_translation

Specifies the default translation for unlisted tools. Valid values are FULL_ON, OFF and UNDETERMINED.

-full_on_tools {*tools*}

Specifies the tools in which PARTIAL_ON is translated to FULL_ON. The tool names are determined by simulation tool and implementation tool doesn't need to do any tool name check for the specified value.

-off_tools {*tools*}

Specifies the tools in which PARTIAL_ON is translated to OFF. The tool names are determined by simulation tool and implementation tool doesn't need to do any tool name check for the specified value.

DESCRIPTION

This command defines the translation of PARTIAL_ON to FULL_ON or OFF for purposes of evaluating the power state of supply sets and power domains. This command does not affect implementation and is intended to be used by simulation tools, to read and write transparently.

EXAMPLES

The following example sets the partial_on_translation to FULL_ON.

```
prompt> set_partial_on_translation FULL_ON \
    -off_tools {OTHER-TOOL}
```

SEE ALSO

create_power_switch(2)

set_path_margin

Specifies a margin to adjust required times for paths in the design.

SYNTAX

status **set_path_margin**
[-setup]
[-hold]
[-rise]
[-fall]
[-from *from_list*
| -rise_from *rise_from_list*
| -fall_from *fall_from_list*]
[-through *through_list*]*
[-rise_through *rise_through_list*]*
[-fall_through *fall_through_list*]*
[-to *to_list*
| -rise_to *rise_to_list*
| -fall_to *fall_to_list*]
[-reset_path]
[-comment *comment*]
[-modes *mode_list*]
[-corners *corner_list*]
[-scenarios *scenario_list*]
margin_value

Data Types

from_list list
rise_from_list list
fall_from_list list
through_list list
rise_through_list list
fall_through_list list
to_list list
rise_to_list list
fall_to_list list
comment string
mode_list list
corner_list list
scenario_list list
margin_value float

ARGUMENTS

-setup

Specifies that setup (maximum delay) calculations are to use the specified *margin_value* for the specified paths. By default, both **-setup** and **-hold** constraints are adjusted.

-hold

Specifies that hold (minimum delay) calculations are to use the specified *margin_value* for the specified paths. By default, both **-setup** and **-hold** constraints are adjusted.

-rise

Specifies that timing adjustment applies only to paths having a rising signal. If neither **-rise** nor **-fall** is specified, both rising and falling delays are affected. Rise refers to a rising value at the path endpoint.

-fall

Specifies that timing adjustment applies only to paths having a falling signal. If neither **-rise** nor **-fall** is specified, both rising and falling delays are affected. Fall refers to a falling value at the path endpoint.

-from *from_list*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, or a pin that has input delay specified. If you specify a clock, all path startpoints related to that clock are affected. If you specify a cell name, one path startpoint on that cell is affected. All paths from these startpoints to the endpoints in the to_list are adjusted by the margin_value. If you don't specify to_list, all paths from from_list are affected.

-rise_from *rise_from_list*

Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-fall_from *fall_from_list*

Same as the **-from** option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-from**, **-rise_from**, and **-fall_from**.

-through *through_list*

Specifies a list of pins, ports, cells and nets through which the paths must pass. The margin value applies only to paths that pass through one of the points in the *through_list*. You can specify **-through** more than one time in a single command invocation. If you specify the **-through** option multiple times, the margin values apply to paths that pass through a member of each *through_list* in the order in which the lists were given. If you use the **-through** option together with the **-from** or **-to** option, the margin adjustment applies only if the **-from** or **-to** conditions are satisfied and the **-through** conditions are satisfied.

-rise_through *rise_through_list*

This option is similar to the **-through** option, but applies only to paths with a rising transition at the specified objects. You can specify **-rise_through** more than one time in a single command invocation.

-fall_through *fall_through_list*

This option is similar to the **-through** option, but applies only to paths with a fall transition at the specified objects. You can specify **-fall_through** more than one time in a single command invocation.

-to *to_list*

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs

related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected. You can use only one of **-to**, **-rise_to**, and **-fall_to**. All paths to the endpoints in the *to_list* are adjusted by margin_value. If you don't specify a *from_list*, all paths to *to_list* are affected. If you specify a clock, all path endpoints related to that clock are affected.

-rise_to *rise_to_list*

Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-fall_to *fall_to_list*

Same as the **-to** option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of **-to**, **-rise_to**, and **-fall_to**.

-reset_path

Specifies that existing point-to-point exception information is to be removed from the specified paths. If used with **-to** only, all paths leading to the specified endpoints are reset. If used with **-from** only, all paths leading from the specified startpoints are reset. If used with **-from** and **-to**, only paths between those points are reset. Only information of the same rise or fall setup or hold type is reset. Using this option is equivalent to using the **reset_paths** command with similar arguments before issuing **set_path_margin**.

-comment *comment*

Associates a string description with the command for tracking purposes. This can be useful for optimization engines to track which flow or engine applied the margin values and hence useful for setting and clearing the values.

-modes *mode_list*

Specifies the scenarios to which the path margin delay value is applied. If this option is given, all scenarios of the given modes are used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-corners *corner_list*

Specifies the scenarios to which the path margin delay value is applied. If this option is given, all scenarios of the given corners and the current mode is used. The **-scenarios** option must not be specified with this option. If both **-modes** and **-corners** are specified, all scenarios whose mode is specified by **-modes** and whose corner is specified by **-corners** are used.

-scenarios *scenario_list*

Specifies the scenarios to which the path margin delay value is applied. The **-modes** or **-corners** option must not be given with this option. Note that when the -scenarios option is specified and clocks are specified for one or more of the -from, -rise_from, -fall_from or -to, -rise_to, -fall_to options, then only scenarios of current_mode, are considered.

margin_value

Specifies the value of the margin by which to adjust the required time for the specified paths. A positive value results in a more restrictive or tighter check. A negative value results in a less restrictive or looser check.

DESCRIPTION

This command adjusts the data required time for specified paths by the specified *margin_value* amount. The required time for any startpoint in *from_list* to any endpoint in *to_list* is adjusted by *margin_value*. A positive margin value results in a more restrictive or tighter check, whereas a negative margin value results in a less restrictive or looser check. Based on the type of timing constraint,

either setup or hold, the margin value is either subtracted from or added to the default data required time.

The **set_path_margin** command is a point-to-point timing exception command, that is, it adjusts the required time of the path and the endpoint slack for the timing paths.

Other point-to-point timing exception commands include the **set_multicycle_path**, **set_min_delay**, and **set_false_path**, **set_max_delay** commands.

If a path satisfies multiple timing exceptions, the following rules are applied to determine which exceptions take effect. Rules referring to **-from** apply equally to **-rise_from** and **-fall_from**, and similarly for the rise and fall options of **-through** and **-to**.

1. If one exception is a **set_false_path** exception, and the other is a **set_path_margin** exception, the **set_false_path** exception takes precedence.

2. If one exception is a **set_path_margin** exception and the other is a **set_max_delay**, **set_min_delay**, or **set_multicycle_path** exception, no conflict occurs and both constraints are honored.

3. If one exception has a **-from** pin or **-from** cell and the other does not, the former takes precedence.

4. If one exception has a **-to** pin or **-to** cell and the other does not, the former takes precedence.

5. If one exception has any **-through** points and the other does not, the former takes precedence.

6. If one exception has a **-from** clock and the other does not, the former takes precedence.

7. If one exception has a **-to** clock and the other does not, the former takes precedence.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenarios** option or the **-corners** and **-modes** options.

EXAMPLES

The following example adjusts the required time to a port named OUT by 12 units for corner C1. The adjustment results in a more restrictive, tighter timing constraint because the margin value is positive.

```
prompt> set_path_margin 12 -to {OUT} -corners C1
```

This example specifies that the required time of all paths from cell FF1 that pass through cell I1 and end at cell FF2 must be adjusted by 15.0 units for corner C2.

```
prompt> set_path_margin 15 -from {FF1} -through {I1} -to {FF2} -corners C2
```

This example specifies that all paths to endpoints clocked by CLK must be adjusted by 10 units for corners C1 and C2.

```
prompt> set_path_margin 10 -to [get_clocks CLK] -corners {C1 C2}
```

SEE ALSO

current_design(2)
report_exceptions(2)
reset_design(2)

reset_paths(2)
set_false_path(2)
set_max_delay(2)
set_min_delay(2)
set_multicycle_path(2)

set_pg_mask_constraint

Sets a PG mask constraint. The constraints includes PG strap width, PG strap alignment mode with colored routing tracks, and so on.

SYNTAX

status **set_pg_mask_constraint**
name
-track_alignment_mode *mode_list*
[-width *value_list*]
[-track_alignment_offset *value_list*]

Data Types

name string
mode_list list
value_list list

ARGUMENTS

name

Specifies the name of the PG mask constraint.

-track_alignment_mode *mode_list*

Specifies the routing track alignment checking mode in this constraint. The argument for **-track_alignment_mode** is one of *aligned* or *inverse*. *aligned* aligns the center of the PG strap with routing track of the same mask color. *inverse* aligns the center of the PG strap with routing track of the opposite mask color. This is a required option.

-width *value_list*

Specifies the valid width values of the PG wires in this constraint. Only PG straps with one of the specified width values can be created. The number of values you specify must be the same as number of alignment modes in "-track_alignment_mode" option. This is not required. By default, any width is valid if this option is not specified.

-track_alignment_offset *value_list*

Specifies the routing track alignment offset in this constraint. The offset value is defined with respect to track, that is, the center of the strap minus the offset value is checked against the routing track based on aligned or inverse modes. The number of values needs to be the same as number of alignment modes in **-track_alignment_mode** option. By default the offset is 0 which means the center of the strap is checked against routing track.

DESCRIPTION

This command sets a PG mask constraint which includes PG strap width, PG strap alignment mode with colored routing tracks, and so on. PG straps are checked against the specified PG mask constraint in wire pattern, mesh pattern or atomic commands.

EXAMPLES

The following example sets a PG mask constraint named "mask_constraint", where allowed PG strap location and width values are {track: aligned, offset: 0.5, width 3}, {track: inverse, offset: -0.5, width 2}, and {track: aligned, offset: 0.2, width 1}.

```
prompt> set_pg_mask_constraint\
   mask_constraint \
   -track_alignment_mode {aligned inverse aligned} \
   -track_alignment_offset {0.5 -0.5 0.2} \
   -width {3 2 1}
```

SEE ALSO

compile_pg(2)
create_pg_strap(2)
create_pg_wire_pattern(2)
create_pg_mesh_pattern(2)
report_pg_mask_constraints(2)
remove_pg_mask_constraints(2)
set_pg_strategy(2)

set_pg_strategy

Specifies the power ground network creation strategy.

SYNTAX

status **set_pg_strategy**
strategy_name
[-core | -design_boundary | -voltage_areas *voltage_areas* |
-polygon {*polygon_area*} | -macros *macro_names* |
-blocks *block_names* | -pg_regions *pg_region_names*]
-pattern *pattern_expr*
[-extension {*extension_spec*}]
[-blockage {*blockage_spec*}]

Data Types

strategy_name	string
voltage_areas	collection
polygon_area	list
macro_names	collection
block_names	collection
pg_region_names	list
pattern_expr	specification
extension_spec	specification
blockage_spec	specification

ARGUMENTS

The options **-core**, **-design_boundary**, **-voltage_areas**, **-polygon**, **-macros**, **-blocks** and **-pg_regions** are mutually exclusive. One and only one of these options needs to be specified.

strategy_name

Specifies the name of the strategy.

-core

Uses the core area as the power ground region.

-design_boundary

Uses the design boundary as the power ground region.

-voltage_areas *voltage_area*

Uses the specified voltage area as the power ground region.

-polygon {*polygon_area*}

Uses the specified polygon area as the power ground region. The polygon area is specified by a list of coordinate points.

-macros {*macro_names*}

Uses the specified macros as the power ground region.

-blocks {*block_names*}

Uses the specified blocks as the power ground region.

-pg_regions *pg_region_names*

Uses the power ground regions created by the **create_pg_region** command as the power ground regions. Multiple regions can be specified in the list *pg_region_names*.

-pattern *pattern_expr*

Specifies the power ground pattern to be instantiated in this strategy. The pattern can be wire, composite, mesh, ring, macro connection, standard cell rail connection, or special pattern. Only one pattern can be specified and this is a required option.

For wire, composite, mesh, connection or special pattern, the strategy instantiates the pattern inside the routing area by repeating the pattern. For macro connection pattern, the strategy instantiates the pattern by applying the pattern to macros inside the routing area. For a standard cell rail connection pattern, the strategy instantiates the pattern by creating standard cell rails inside the routing area. The pattern expression *pattern_expr* is specified as follows:

{name: *pattern_name*}
{nets: *real_net_list*}
{parameters: *var_list*}
{offset: {*x_offset y_offset*}}
{offset_start: boundary | {*x y*}}

The **name** keyword followed by *pattern_name* specifies the name of the pattern. For standard cell connection pattern, if the specified pattern name is **std_cell_rail** and this pattern does not exist, a default standard cell rail pattern **std_cell_rail** is created.

The **nets** keyword followed by *real_net_list* specifies the list of net names to be mapped into the symbolic list of nets in *pattern_name*.

The **parameters** keyword followed by *var_list* specifies the parameter list. The parameter list is passed as arguments into the pattern *pattern_name*.

The **offset** keyword followed by {*x_offset y_offset*} specifies the horizontal and vertical offsets when creating the first pattern. By default, the offset is 0 for both horizontal and vertical offsets. Offset does not apply to the scattered pin macro connection pattern or the standard cell rail pattern.

The **offset_start** keyword specifies the offset reference point in routing area for pattern offsets. The **boundary** keyword refers to the lower-left point of the routing area bounding box. Any other point can also be specified as the reference point by coordinate {*x y*}. By default, boundary is used. Offset start does not apply to scattered pin macro connection pattern or standard cell rail pattern.

For a ring pattern, the strategy instantiates the pattern around the routing area. The pattern expression *pattern_expr* is specified as follows:

{name: *pattern_name*}
{nets: *real_net_list*}
{parameters: *var_list*}
{offset: {*x_offset y_offset*}}
{skip_sides: *id_list*}
{side_offset: *side_offset_spec*}

The **pattern** keyword followed by *pattern_name* specifies the pattern name.

The **nets** keyword followed by *real_net_list* specifies the net name list. The nets are mapped into the symbolic list of nets in *pattern_name*.

The **parameters** keyword followed by *var_list* specifies the parameter list which will be passed as arguments into the pattern *pattern_name*.

The **offset** keyword followed by {*x_offset y_offset*} specifies the horizontal and vertical offsets when creating ring segments around the routing area. The offset is the distance between innermost ring to the routing area boundary. By default, the offset is 0 for both horizontal and vertical offsets.

The **skip_sides** keyword specifies the edge ID list to be skipped. *id_list* specifies the edge ID list to be skipped for ring creation. Each edge is indexed by a unique number. Edge numbering starts from 1 at the leftmost vertical edge. If there is more than one leftmost edge, the bottommost edge is the starting edge. The edge number increases by 1 as you proceed around the shape in the clockwise direction. By default, no edge is skipped for ring creation.

The **side_offset** keyword specifies the offset for each edge. Each edge is indexed by a unique number. Edge numbering starts from 1 at the leftmost vertical edge. If there is more than one leftmost edge, the bottommost edge is the starting edge. The edge number increases by 1 as you proceed around the shape in the clockwise direction. *side_offset_spec* specifies the offset for each edge as a pair of **edge_id** and **offset** keywords. An example is as follows:

{{side: *edge_id*}{offset: *edge_offset*}}

where *side* and *offset* specify the edge ID and edge offset respectively. The offset unit is defined in the technology file. You can specify negative offsets to shrink the region. If you do not specify offsets for all edges, the offset for the remaining edges is either 0 or the uniform offset for all edges set by **-offset** option.

-extension {*extension_spec*}

Specifies how a group of power and ground nets should extend beyond the power grid. You can specify multiple extension specifications separated by braces. Each specification contains the keywords **nets**, **side**, **layers**, **direction** and **stop** as follows:

{{nets: *nets*}
{side : *id_list*}
{layers : *layer_list*}
{direction: *directions*}
{stop: *stop_target*}}

The net names following the *nets* keyword are a subset of the nets specified by **-nets** in *pattern_expr* from **-pattern** option. If you do not specify the **nets** keyword, the extension applies to all the nets in the strategy.

The **side** keyword specifies the ring segment IDs or macro side IDs by *id_list*. Each ring edge or macro boundary is indexed by a unique number. Edge numbering starts from 1 at the leftmost vertical edge. If there is more than one leftmost edge, the bottommost edge is the starting edge. The edge number increases by 1 as you proceed around the shape in the clockwise direction. If not specified, the extension applies to all ring edges or all macro boundary sides. This keyword can be only applied to ring or macro connection type pattern.

The **direction** keyword specifies the extension direction. *directions* contains one or more of the letters "L", "R", "T" or "B"; "L" refers to left, "R" refers to right, "T" refers to top, and "B" refers to bottom. If this entry is not specified, the extension applies to all directions.

The **layers** keyword specifies the extension specifications only applied to the particular layer list by *layer_list*, which contains a list of routing layer names. If this keyword is not specified, the extension applies to all layers.

The specification following the **stop** keyword can be a distance specified in microns, a collection or a list of PG pins, or one of the following keywords: **core_boundary**, **first_target**, **innermost_ring**, **outermost_ring**, **pad_ring**, **half_space**, **design_boundary**, and **design_boundary_and_generate_pin**. **core_boundary** specifies the core area boundary. **first_target** specifies the first wire segment of the same net. **collection_of_pins** specifies the collection of pins to be extended to. **innermost_ring** specifies the innermost ring of the same net, **outermost_ring** specifies the outermost ring of the same net, **pad_ring** specifies the pad ring segment or pad ring pins of the same net, **half_space** specifies half spacing rule to the specified region of the strategy. and **design_boundary** specifies the boundary of the current design. **design_boundary_and_generate_pin** specifies extension to the

design boundary and generating pins. If you specify a distance, the shape extends by the distance from the boundary of the routing region if the pattern is wire pattern, composite pattern, mesh pattern, or standard cell connection pattern, or from the corner of ring segments for ring pattern, or from macro pins for macro connection pattern. You can also make the shapes to extend to a pre-specified set of macro pins. There are two ways to specify the set of pins, use a list of pin names or a collection of pins. For this stop option to work, both nets and layers options need to be specified as well.

Collection of pins example set gnd_pins [get_pins -all -of_objects [get_cell $powerSwitches] -filter "name==GND"] set_pg_strategy -extension {{{nets: GND} {stop: $gnd_pins} ...}...}...

List of pin names example set gnd_pinNames "GND0 GND1 GND2 ..." set_pg_strategy -extension {{{nets: GND} {stop: $gnd_pinNames} ...}...}...

-blockage {*blockage_spec*}

Specifies the routing blockage in the power ground network. You can specify multiple blockages within the braces, one blockage per group. Blockages are separated by braces and contain the keywords **nets:**, **layers:**, and a target specification as follows:

{{nets: *nets*} {layers: *layers*} {*target*}

where *target* is a voltage area, block, macro, macro with hard keep-out-margin (KOM), PG region, hard placement blockage, or polygon that is specified as follows:

{voltage_areas : *voltage_areas* | macros : *macro_names* |
macros_with_keepout : *macro_names* | placement_blockages : all |
polygon : {*polygon_area*} | blocks : *blocks* |
pg_regions : *region_list* }

The net names following the *nets* keyword are a subset of the nets specified by **-nets** in *pattern_expr* from **-pattern** option. If you do not specify the **nets** keyword, the blockage applies to all the nets in the strategy. The layer names following the **layers** keyword are the routing layers on which power or ground straps are blocked. If you do not specify the **layers** keyword, the blockage applies to all routing layers. Use one of the **voltage_areas**, **macros**, **polygon**, **blocks** or **pg_regions** keywords to represent the blockage area. *polygon_area* is specified as a list of coordinate points.

DESCRIPTION

This command specifies the strategy for creating power ground network elements. The command specifies the power ground pattern, power ground nets, power ground routing area, the net extension strategy and the blockage area.

For wire, composite, mesh and special patterns, the strategy instantiates the pattern inside the routing area by repeating the pattern. For a ring pattern, the strategy instantiates the pattern around the routing area. For macro connection pattern, the strategy instantiates the pattern by applying the pattern to macros inside the routing area. For standard cell rail connection pattern, the strategy instantiates the pattern by creating standard cell rails inside the routing area. Blockage and extension can be further specified.

EXAMPLES

The following example defines a power ground strategy s1 for a composite pattern comp_pattern. Parameters 5 and 3 are passed into the pattern for wire width and spacing respectively. The routing area is inside the voltage area VA1. The real net names are VDD and VSS. Both nets extend to the outermost VDD and VSS rings. Macro hm1 inside this voltage area is treated as blockage for VDD but not for VSS.

```
prompt> set_pg_strategy \
```

```
  s1 \
  -voltage_areas {VA1} \
  -pattern {{name: comp_pattern}{nets:{VDD VSS}}{parameters:{5 3}}}\
  -blockage {{nets: VDD}{macros: rf0/x0/t_al}}\
  -extension {stop: outermost_ring}
```

The following example defines a power ground strategy s2 for a ring pattern ring_pattern. Parameters 4, 1, 5, 2 are passed into the pattern for horizontal width, spacing, and vertical width, spacing respectively. The ring is around the routing area defined by power ground region r1. The net names are Vdd and Vss. Horizontal and vertical offsets are 10 and 20 respectively. The ring segment with edge ID 1 is extended in the top direction to the pad ring.

```
prompt> set_pg_strategy \
  s2 \
  -pg_regions {r1} \
  -pattern {{name: ring_pattern}{nets:{Vdd Vss}} \
        {parameters:{4 1 5 2}}{offset:{10 20}}} \
  -extension {{side: 1}{direction: T}{stop: pad_ring}}
```

The following example defines a power ground strategy s3 for a ring pattern ring_pattern. The ring is around the routing area defined by power ground region r1. The net names are Vdd and Vss. Horizontal and vertical offsets are 10 and 20 respectively. The ring segment with edge IDs 1 and 2 is extended in the both bottom and right directions to the design boundary with pin created, and edge ID 3 in the top direction to design boundary with pin created.

```
prompt> set_pg_strategy \
  s3 \
  -pg_regions {r1} \
  -pattern {{name: ring_pattern}{nets:{Vdd Vss}} \
        {offset:{10 20}}} \
  -extension {{{side: 1 2}{direction: B R }{stop: design_boundary_and_generate_pin}} \
         {{side: 3}{direction: T}{stop: design_boundary_and_generate_pin}}}
```

The following example defines a power ground strategy s4 for a macro connection pattern macro_pattern. The nets are Vdd and Vss. The macros to be connected are hm1 and hm2. For the pins at boundary sides indexed by 1, 2 and 3, the extension stops at the first target. For the pins at the side indexed by 4, the extension stops at the innermost ring.

```
prompt> set_pg_strategy \
  s4 \
  -macros "$macro1 $macro2" \
  -pattern {{name: macro_pattern}{nets:{Vdd Vss}}} \
  -extension {{{side: 1 2 3}{stop: first_target}} \
          {{side: 4}{stop: innermost_ring}}}
```

The following example defines a power ground strategy s5 for a standard cell rail connection pattern std_pattern. The nets are Vdd and Vss. The routing area is the core area. The routing layers are M1 and M2 as parameters passed into the pattern. The rails at layer M2 are extended out of the core area by 10 um.

```
prompt> set_pg_strategy \
  s5 \
  -core \
  -pattern {{name: std_pattern}{nets: {Vdd Vss}}{parameters: {M1 M2}}}\
  -extension {{stop: 10}{layers: M2}}
```

MORE EXAMPLES

For more example, please start GUI and invoke the following command. **gui_show_task -task "Design Planning:PG Planning-**

>Examples->Overview"

SEE ALSO

compile_pg(2)
create_pg_composite_pattern(2)
create_pg_macro_conn_pattern(2)
create_pg_mesh_pattern(2)
create_pg_region(2)
create_pg_ring_pattern(2)
create_pg_special_pattern(2)
create_pg_std_cell_conn_pattern(2)
create_pg_wire_pattern(2)
remove_pg_strategies(2)
report_pg_strategies(2)
set_pg_strategy_via_rule(2)
set_pg_via_master_rule(2)

set_pg_strategy_via_rule

Specifies the via rule between strategies and collections of existing shapes.

SYNTAX

status **set_pg_strategy_via_rule**
rule_name
-via_rule *via_rule_spec*

Data Types

rule_name string
via_rule_spec specification

ARGUMENTS

rule_name

Specifies the name of the via rule.

-via_rule *via_rule_spec*

Specifies the via rule *via_rule_spec* between strategies, and also between the strategies and existing shapes. The via rule defines the intersection locations and the via at the locations. Note that *-via_rule* defined in **create_pg_mesh_pattern** and **create_pg_composite_pattern** only applies to the intersections of wires in the same pattern, and do not apply to the intersection of wires of different patterns. The *-via_rule* defined in **create_pg_mesh_pattern** and **create_pg_composite_pattern** do not also apply to the intersection of pattern wires and existing wires.

There are three methods to define the intersection locations in the specification *via_rule_spec*, where the syntax is defined as follows:

{**intersection** : all} {*via_def*} |
{**intersection** : adjacent} {*via_def*} |
list_of_filter_rules

The first method is "{**intersection** : all}", where **intersection** is the keyword and **all** refers to all intersections between any two orthogonal shapes including straps, rings, standard cell rails created by strategies or any existing shapes in any different layers.

The second method is "{**intersection** : adjacent}", where **intersection** is the keyword and **adjacent** refers to all intersections between any two orthogonal shapes only in adjacent layers.

The third method is to define the intersections based on two sets of shapes using *list_of_filter_rules*, which contains a list of filtering rule. Each filtering rule is specified inside a curly brace pair. The syntax of each filtering rule is as follows:

{{*set_1*}{*filter_1*}}{{*set_2*}{*filter_2*}}{*via_def*} |
{**intersection** : undefined}{*via_def*}

where *set_1* refers to the first set of shapes and *filter_1* refers to the filtering operations for this set. *set_2* and *filter_2* refer to the second set of shapes and the filtering operations for the second set, respectively. *set_1* or *set_2* is defined as follows:

{**strategies** : *strategy_name_list*} |
{**existing** : *shape_type* } |
{**macro_pins** | **terminals** : all | *pin_names* }
{**tags**: *{tag_names}* }

where **strategies** is the keyword for strategies, **existing** is the keyword for existing PG wires, **macro_pins** is the keyword for macro pins and **terminals** is the keyword for terminals. Strategy names are specified by *strategy_name_list*. Existing wires are specified by shape type *shape_type*. Valid *shape_type* options are: ring, strap, macro_conn, follow_pin and std_conn. The keywords ring, strap, follow_pin, std_conn, and macro_conn specify ring-type, strap-type, follow pin type, standard cell connection-type, macro pin connection-type and respectively. Macro pins or terminals could be specified by the all keyword, or by a list of pin names *pin_names*. Tags could be specified by list of tag names. The filtering operations of strategy shapes, existing shapes and tags are defined as follows:

{**nets** : *net_list*}
{**layers** : *layer_list*}
{**width** : {*lower_w upper_w*}}

nets, **layers**, **width** keywords can be used to specify the wire net names by *net_list*, wire layer names by *layer_list* or width range by {*lower_w upper_w*}.

The filtering operations of macro pins are defined as follows:

{**nets** : *net_list*}
{**layers** : *layer_list*}
{**width** : {*lower_w upper_w*}}
{**height** : {*lower_h upper_h*}}
{**width_height_ratio** : {*lower_r upper_r*}}

height and **width_height_ratio** refer to pin height and width height ratio respectively.

By specifying two sets of shapes by strategy names, existing shapes or macro pins, and certain filtering operations by net names, layer names, wire width ranges or types, the via locations are the intersections between these two subsets of shapes. For those intersections which do not belong to any filtering rule, the via definition can be specified by

{**intersection** : undefined}{*via_def*}

where **intersection** is the keyword, **undefined** refers to the remaining intersections which do not belong to any filtering rule. By default, vias are only created at the intersections which belong to a filtering rule, unless otherwise specified. Multiple via filtering rules, such as intersection location and via definition, can be defined and separated by curly braces where each filter rule is in one curly brace pair.

With the intersection definition, the via specification {*via_def*} is in the following format:

{**via_master** : *via_master_list*}

where **via_master** is the keyword for both via masters and via master rules. *via_master_list* specifies the list of via masters. Via masters include the via contact code defined in the technology file or user-defined via cells. *via_rule_list* is a list of PG via rules. Via master rules are defined by **set_pg_via_master_rule** command. **NIL** can be used as a keyword to indicate that no via is created. **default** can be used to describe the default vias in the specified location. For each specified intersection, vias are created based on the specified via masters or via master rules. If **NIL** is not specified in the list, the default contact code is used to complete a stacked via or to replace a via with DRCs when applicable.

By default, the default via defined in the technology file is created at each intersection of orthogonal wires in different layers.

DESCRIPTION

Specifies the via rule between two different strategies or a strategy and existing objects. The via rule can be used with the **compile_pg** command when creating the power and ground network.

EXAMPLES

The following example specifies a via rule with name r1 where vias will be created only between adjacent layers.

```
prompt> set_pg_strategy_via_rule r1 \
   -via_rule {{intersection: adjacent}{via_master: default}}
```

The following example specifies a via rule with name r2 which includes a list of filtering rules. Vias VIA23 will be created at the intersection between the shapes from strategy s1 at layer M2 and the shapes from strategy s2 at layer M3. VIA34 will be created at the intersection between the shapes from strategy s2 at layer M3 and all existing ring shapes. Vias will not be created at other intersections between strategy shapes and existing shapes.

```
prompt> set_pg_strategy_via_rule r2 \
  -via_rule { \
      {{{strategies: s1}{layers: M2}}{{strategies: s2}{layers: M3}} \
           {via_master: VIA23}} \
      {{strategies: s2}{existing : ring} \
           {via_master : VIA34}} \
      {{intersection: undefined}{via_master: NIL}} \
  }
```

The following example specifies a via rule with name r3 which includes a list of filtering rules. Vias VIA23 and VIA34 will be created at the intersection between parallel shapes from strategy s1 at layer M2 and the shapes from strategy s2 at layer M4. Vias will not be created at all other intersections.

```
prompt> set_pg_strategy_via_rule r3 \
    -via_rule { \
      { {{strategies: s1} {layers: M2}} \
        {{strategies: s2} {layers: M4}}{via_master: {VIA23 VIA34}} \
        {between_parallel: true} } \
      { {intersection: undefined}{via_master: NIL} } \
  }
```

The following example specifies a via rule with name r4 which includes a list of filtering rules. Vias VIA23 and VIA34 will be created at the intersection between parallel shapes from strategy s1 at layer M4 and the shapes from existing std rail at layer M2. Vias will not be created at all other intersections.

```
prompt> set_pg_strategy_via_rule r4 \
    -via_rule { \
      {{{strategies: s1} {layers: M4}}{{existing: std_conn} {layers: M2}} \
        {via_master: {VIA23 VIA34}}{between_parallel: true}} \
       {{intersection: undefined}{via_master: NIL}} \
  }
```

The following example specifies a via rule with name r5 which includes a list of filtering rules. Default vias will be created at the intersection between shapes from strategy s1 and the existing shapes with tag name std_rail. Vias will not be created at all other intersections.

```
prompt> set_pg_strategy_via_rule r5 \
```

```
  -via_rule { \
    {{strategies: s1}{tags: {std_rail}}{via_master: {default}}} \
     {{intersection: undefined}{via_master: NIL}} \
}
```

The following example specifies a via master "staple_via" with contact code VIA12C, cuts of 2 columns and 1 row, horizontal pitch 0.2. The via rule rail_rule includes a list of filtering rules. Via master staple_via will be created at the intersection between parallel shapes from strategy std_1 and the shapes from existing straps at layer M2. Vias will not be created at all other intersections.

```
prompt> set_pg_via_master_rule staple_via -contact_code VIA12C \
  -via_array_dimension {2 1} -allow_multiple {0.2 0}
prompt> create_pg_std_cell_conn_pattern rail_pattern -layers M1
prompt> set_pg_strategy std_1 -core \
  -pattern {{pattern: rail_pattern} {nets: VDD VSS} }
prompt> set_pg_strategy_via_rule rail_rule \
  -via_rule { \
    { {strategies: {std_1}} \
      {{existing: strap}{layers: M2}} \
      {via_master:staple_via} {between_parallel: true} } \
    { {intersection: undefined}{via_master: NIL} }}
prompt> compile_pg -strategies {std_1} -via_rule {rail_rule}
```

MORE EXAMPLES

For more example, please start GUI and invoke the following command. **gui_show_task -task "Design Planning:PG Planning->Examples->Overview"**

SEE ALSO

compile_pg(2)
remove_pg_strategy_via_rules(2)
report_pg_strategy_via_rules(2)
set_pg_strategy(2)
set_pg_via_master_rule(2)

set_pg_via_master_rule

Creates a new via master rule for the power and ground network. The via rule is defined by the contact code, location of contact code inside the intersection box, cut spacing, multiplication, and so on.

SYNTAX

status **set_pg_via_master_rule**
rule_name
[-contact_code *via_contact_code*]
[-via_array_dimension {*column row*}]
[-offset_start center | lower_left]
[-offset {*x_offset y_offset*}]
[-orient R0 | R90]
[-allow_multiple {*x_pitch y_pitch*}]
[-cut_spacing {*x_spacing y_spacing*}]
[-via_site_ratio {*x_ratio y_ratio*}]
[-track_alignment *alignment_mode*]
[-cut_mask mask_one | mask_two | mask_three]
[-snap_reference_point {*x y*}]

Data Types

rule_name string
via_contact_code string
column integer
row integer
x_offset float
y_offset float
x_pitch float
y_pitch float
x_spacing float
y_spacing float
x_ratio float
y_ratio float
alignment_mode string
x float
y float

ARGUMENTS

rule_name

Specifies the name of the power ground via rule. This option is required.

-offset {*x_offset y_offset*}

Specifies the horizontal (*x_offset*) and vertical (*y_offset*) values to use when creating a contact code inside the intersection box. The units are um. The offset is applied with respect to the center of the intersection box. By default, the offset is 0.

-offset_start center | lower_left

Specifies the offset reference point in the intersection box. Keyword **center** refers to the center of the intersection box. Keyword **lower_left** refers to the lower-left corners of the intersection box.

-contact_code *via_contact_code*

Specifies the via master or contact code used to create the via rule. By default, the tool uses the default contact codes.

-via_array_dimension {*column row*}

Specifies the via array dimension size by integer values for *column* and *row*. This option can only be applied to a simple symmetric contact code. The tool creates the via array based on the specified contact code. If the contact code is not symmetric, this option is ignored. By default, *column* and *row* are calculated based on the size of the intersection, and the tool fills the intersection with via cuts.

-orient R0 | R90

Specifies the orientation of the contact code by keywords **R0** and **R90** respectively. By default, the via has orientation R0 and is not rotated.

-cut_mask mask_one | mask_two | mask_three

Specifies the cut mask for this via master rule by keywords **mask_one**, **mask_two** or **mask_three**. By default, the via cut mask is not colored. Via metal enclosure masks follow PG wire masks.

-allow_multiple {*x_pitch y_pitch*}

Specifies the horizontal (*x_pitch*) and vertical (*y_pitch*) values when filling an intersection with multiple repetitions of the contact code or via array. The unit is defined in the technology file. If the via array is created with the **-via_array_dimension** option, the via array is repeated. Otherwise, the contact code defined by the **-contact_code** option is repeated. By default, the contact code or via array is not repeated.

-cut_spacing {*x_spacing y_spacing*}

Specifies the horizontal (*x_spacing*) and vertical (*y_spacing*) values between cuts in a simple array via. The spacing values are ignored if smaller than the minimum cut spacing. The unit of cut spacing is in um. By default, the default cut spacing is used.

-via_site_ratio {*x_ratio y_ratio*}

Specifies the via site horizontal and vertical ratio by *x_ratio* and *y_ratio*. The valid range for ratio is from 0.0 to 1.0 while 0.0 is invalid. By default, the ratio is 1.0 and the whole intersection is used for via creation. When specified, the intersection box is shrunk by the specified ratio for via creation.

-track_alignment *alignment_mode*

Specifies the track alignment option for this via master rule. Valid values are **track**, **half_track**, **top_track_bottom_half_track**, **top_half_track_bottom_track**, **top_track_only**, **top_half_track_only**, **bottom_track_only**, and **bottom_half_track_only**.

By default vias created by this rule do not align to routing track or half routing track. The top and bottom of a via can be aligned to track or half-track respectively. The alignment can be applied to both of layers or just one of the top and bottom layers.

-snap_reference_point {*x y*}

Specifies the reference point for repeating vias. This option needs to be used together with the "-allow_multiple" option for repeating vias. When specified, the origin of the first via or the lower-left via created inside the intersections with this via master rule will be snapped to the specified point, or the specified point plus a multiple of pitch values if the reference point is outside the

intersection. The "-offset_start" or "-offset" options cannot be used together with this option. If not specified, repeating vias will honor offset options or get evenly distributed inside intersection bounding boxes.

DESCRIPTION

Specifies the power ground via rules such as the location of contact code inside the intersection box, orientation and multiplication etc. The PG via rule can be used during pattern definition and when creating the power and ground network.

EXAMPLES

The following example defines a power ground via rule with the name via_rule_1 using contact code VIA34. The offsets are 3 um and 4 um with respect to the center of the intersection. The via array size is 5x5.

```
prompt> set_pg_via_master_rule \
  via_rule_1 -contact_code VIA34 \
  -offset {3 4} -via_array_dimension {5 5} \
```

MORE EXAMPLES

For more example, please start GUI and invoke the following command. **gui_show_task -task "Design Planning:PG Planning->Examples->Overview"**

SEE ALSO

compile_pg(2)
create_pg_composite_pattern(2)
create_pg_macro_conn_pattern(2)
create_pg_ring_pattern(2)
remove_pg_via_master_rules(2)
report_pg_via_master_rules(2)

set_pin_budget_constraints

Specifies how to allocate budgeted delays at selected pins.

SYNTAX

int **set_pin_budget_constraints**
[-from_percent *percent*]
[-from_delay *delay*]
[-to_percent *percent*]
[-to_delay *delay*]
[-internal_percent *percent*]
[-internal_delay *delay*]
[-none]

[-early_boundary *boundary_name*]
[-late_boundary *boundary_name*]

[-frozen]
[-not_frozen]
[-reset]

[-same_as_mib]
[-not_same_as_mib]
[-same_as_pin *port_name*]
[-not_same_as_pin]
[-same_as_modes modes]
[-not_same_as_modes]
[-same_as_feedthrough]
[-not_same_as_feedthrough]

[-inputs]
[-outputs]
[-feedthrough]
[-internal]
[-from_clock *clock*]
[-from_clock_rise]
[-from_clock_fall]
[-to_clock *clock*]
[-to_clock_rise]
[-to_clock_fall]

[-modes *mode_list*]
[-all_modes]

[-all_pins | -all | *pin_list*]

Data Types

percent float (percent value between 0 and 100)
delay float (delay value in input time units)
margin float
boundary_name string
port_name string
clock string
mode_list collection
pin_list collection of pins

ARGUMENTS

-from_percent *percent*

Specifies the percent of a budget path which should be allocated between the start of the path and the budgeted pin. Any delay remaining in a budget path will be allocated between the budgeted pin and the end of the path. This option cannot be specified with any other delay or percent specification.

-from_delay *delay*

Specifies the delay which should be allocated between the start of the path and the budgeted pin. Any delay remaining in a budget path will be allocated between the budgeted pin and the end of the path. This option cannot be specified with any other delay or percent specification.

-to_percent *percent*

Specifies the percent of a budget path which should be allocated between the budgeted pin and the end of the path. Any delay remaining in a budget path will be allocated between the start of the path and the budgeted pin. This option cannot be specified with any other delay or percent specification.

-to_delay *delay*

Specifies the delay which should be allocated between the budgeted pin and the end of the path. Any delay remaining in a budget path will be allocated between the start of the path and the budgeted pin. This option cannot be specified with any other delay or percent specification.

-internal_percent *percent*

Specifies the percent of a budget path which should be allocated to the inside of a pin's block. Any delay remaining in a budget path will be allocated outside of the block. This option automatically sets the **-internal** option, so a budget with this option will not be applied to feedthrough paths. Setting **-input -internal_percent** is equivalent to setting **-input -internal -to_percent**. Setting **-output -internal_percent** is equivalent to setting **-output -internal -from_percent**. This option cannot be specified with any other delay or percent specification.

-internal_delay *delay*

Specifies the delay which should be allocated to the inside of a pin's block. Any delay remaining in a budget path will be allocated outside of the block. This option automatically sets the **-internal** option, so a budget with this option will not be applied to feedthrough paths. Setting **-input -internal_delay** is equivalent to setting **-input -internal -to_delay**. Setting **-output -internal_delay** is equivalent to setting **-output -internal -from_delay**. This option cannot be specified with any other delay or percent specification.

-none

Specifies that no budget constraint should be generated for a pin.

-early_boundary *boundary_name*

Specifies that a previously defined *budget boundary* should be associated with the given pins. Budget boundaries are defined using the **set_boundary_budget_constraints** command. They control how **set_driving_cell** and **set_load** are applied in the final budget. This option controls what boundary constraints should be used for early (hold time) budgets. Budget boundaries are applied to all timing modes, not just the current mode.

-late_boundary *boundary_name*

Specifies that a previously defined *budget boundary* should be associated with the given pins. Budget boundaries are defined using the **set_boundary_budget_constraints** command. They control how **set_driving_cell** and **set_load** are applied in the final budget. This option controls what boundary constraints should be used for late (setup time) budgets. Budget boundaries are applied to all timing modes, not just the current mode.

-frozen

Specifies that the budgeted constraint for a pin should not be changed by this command or by the automatic budget calculation of the **compute_budget_constraints** command.

-not_frozen

Reverses the effect of the **-frozen** option and allow a pin's budget to be changed.

-reset

Removes any budget specification previously assigned to a pin by this command or by the **compute_budget_constraints** command.

-same_as_mib

Forces the budget constraint to be the same across multiply instantiate block instances. If a pin has this attribute set, it will get the same budget constraint as an identical pin on another multiply instantiated instance that also has this attribute set. By default, all internal budget constraints have this attribute set. Feedthrough constraints do not have this attribute set by default. You must also specify **-internal** or **-feedthrough** when you use this option.

-not_same_as_mib

Reverses the effect of the **-same_as_mib** option. Only the specified pins are affected by this option. Note that internal constraints have **same_as_mib** set by default, and you will need to use the **-not_same_as_mib** option to change the default for internal constraints.

-same_as_pin *port_name*

Forces the budget constraint on specified pins to be the same as the constraint on the named port specified by *port_name*. The named port must be found on the same instance as the specified pins. For example, to give all pins in a bus the same constraint, you might set all of the pins U1/U2/bus[*] to be the same as port bus[0]. See the EXAMPLES section for an example of the **-same_as_pin** option. You must also specify **-internal** or **-feedthrough** when you use this option.

-not_same_as_pin

Reverses the effect of the **-same_as_pin** option for the specified pin. Only the specified pins are affected by this option.

-same_as_modes *modes*

Forces the budget constraint on specified pins to be the same as the constraint on the same pin in another modes. See the EXAMPLES section for an example of the **-same_as_modes** option. You must also specify **-internal** or **-feedthrough** when you use this option.

-not_same_as_modes

Reverses the effect of the **-same_as_modes** option. Pins that are not specified will not be affected.

-same_as_feedthrough

Forces the budget constraints for the two paths to be the same for pins that have both internal paths and feedthrough paths. This option will probably be rarely used. You must also specify **-internal** when you use this option.

-not_same_as_feedthrough

Reverses the effect of the **-same_as_feedthrough** option. Only the specified pins are affected by this option.

-inputs

Specifies that the applied budget parameter should only apply to paths that pass through input pins of blocks. Budget parameters for other pins are not changed. For inout pins, the budget parameter is only applied to paths in the input direction.

-outputs

Specifies that the applied budget parameter should only apply to paths that pass through output pins of blocks. Budget parameters for other pins are not changed. For inout pins, the budget parameter will only be applied to paths in the output direction.

-feedthrough

Specifies that the applied budget parameter should only apply to paths that feed through the specified block pin. That is, there must be a combinational path between a block input and block output. The budget for paths through input pins that end in the block will not be affected. The budget for paths that start in the block and pass through output pins will not be affected.

-internal

Specifies that the applied budget parameter should only apply to paths that start or end in the given pin's block. That is, there must either be a path through input pins that ends in the block, or there must be a path that starts in the block and passes through output pins. The budget for combinational paths between a block input and block output will not be affected.

-from_clock ***clock***

Specifies that the applied budget parameter should only apply to paths with the specified launching clock. Any timing path whose clock does not match the specified clock will be untouched. Previous budget parameters for other clocks, if any, will be maintained. The specified clock will come from the current mode. You must not specify this option together with the **-modes** or **-all_modes** options.

-from_clock_rise

If this option is specified along with **-from_clock**, only paths which trigger on the rising edge of the clock (at the clock source) will be affected. The default is that the clock edge does not matter.

-from_clock_fall

If this option is specified along with **-from_clock**, only paths which trigger on the falling edge of the clock (at the clock source) will be affected. The default is that the clock edge does not matter.

-to_clock ***clock***

Specifies that the applied budget parameter should only apply to paths with the specified capturing clock. Any timing path whose clock does not match the specified clock will be untouched. Previous budget parameters for other clocks, if any, will be maintained. The specified clock will come from the current mode. You may not specify this option along with the **-modes** or **-all_modes** options.

-to_clock_rise

If this option is specified along with **-to_clock**, only paths which trigger on the rising edge of the clock (at the clock source) will be affected. The default is that the clock edge does not matter.

-to_clock_fall

If this option is specified along with **-to_clock**, only paths which trigger on the falling edge of the clock (at the clock source) will be

affected. The default is that the clock edge does not matter.

-modes *mode_list*

By default, budgets are only applied in the current mode. Use this option to specify that the budget should be applied in the specified mode or modes.

-all_modes

Applies the budget to all modes. By default, budgets are only applied in the current mode.

-all_pins

Applies the specified budget to all pins of all budget blocks. The budget will only be applied to the current mode, unless you use the **-modes** or **-all_modes** options. The budget will not be applied for pins that are filtered out by the **-from_clock**, **-to_clock**, **-internal**, **-feedthrough**, **-inputs**, or **-outputs** options.

-all

Applies the specified budget to all pins of all budget blocks in all modes. However, the budget will not be applied for pins that are filtered out by the **-from_clock**, **-to_clock**, **-internal**, **-feedthrough**, **-inputs**, or **-outputs** options. This option is equivalent to specifying **-all_pins -all_modes**.

pin_list

Specifies a list of pin names on budget blocks of your design where a budget should be applied. If a specified pin is not on a budgeted block (as specified by the **set_budget_options** command), a warning is issued. The budget only applies to the current mode, unless you use the **-modes** or **-all_modes** options. The budget is not changed for pins that are filtered out by the **-from_clock**, **-to_clock**, **-internal**, **-feedthrough**, **-inputs**, or **-outputs** options.

DESCRIPTION

This command specifies how to allocate budgeted delay at selected pins. To set a constraint, you must choose one of the budget style options: **-from_delay**, **-to_delay**, **-internal_delay**, **-from_percent**, **-to_percent**, **-internal_percent**, **-none**, or **-reset**.

The budget is applied to the specified pins or all budget pins if **-all_pins** or **-all** is specified. The budget is only applied in the current timing more, unless you specify **-modes**, **-all_modes** or **-all**.

This command provides filters to select only specific pins out of the possible choices. The possible filter options are: **-from_clock**, **-from_clock_fall**, **-to_clock**, **-to_clock_fall**, **-feedthrough**, **-internal**, **-inputs**, and **-outputs**. If more than one budget parameter is applied to the same pin, the more specific parameter is applied. For example, if all paths through a pin (which is less specific) have a **-from_percent** constraint of 30%, but paths with **-to_clock sysclk** (which is more specific) have a **-from_percent** setting of 40%, the 40% constraint is applied wherever possible. The order in which constraints are applied in your script is not important.

Two important filters are **-internal** and **-feedthrough**. Feedthrough budgets only apply to paths that feed through the given pin's block. That is, there must be a combinational path between a block input and block output. Internal budgets only apply to paths that start or end in the given pin's block. That is, there must either be a path through input pins that ends in the block, or there must be a path that starts in the block and passes through output pins.

You can use this command to create "same_as" links between different constraints. The possible same_as options are: **-same_as_mib**, **-not_same_as_mib**, **-same_as_pin port_name**, **-not_same_as_pin**, **-same_as_modes modes**, **-not_same_as_modes**, **-same_as_feedthrough**, and **-not_same_as_feedthrough**. By using these options, you can force the budget constraints for many pins to be the same as other pins. For example, you can set all of the bits of a bus to the same constraint. Or you can apply the same budget constraints to all instances of a MIB (multiply instantiate block). After a "same_as" link has been made, if you set a **-from_***, **-to_*** or **-internal** budget constraint on one pin of the set, the other pins will also get the same constraint.

The same_as controls apply both to manually-set budgets and to automatically-generated budgets. However, the same_as controls do **not** apply to budget constraints set by **-from_clock** or **-to_clock**. By default, internal budgets have *set_same_as_mib* set. Feedthrough budgets do not.

Delay allocations in a budget are based on "budget paths". The total delay in a budget path depends on many things, such as clock period, clock uncertainty, clock latency, and various reserved budget margins. See the **set_budget_options** and **set_latency_budget_constraints** commands for a description of budget margins. You can use the **report_budget** command to see the total delay along various budget paths.

Budget boundary constraints (**set_driving_cell** and **set_load**) can be associated with budgeted pins by using the **-early_boundary** and **-late_boundary** options. Boundary constraints apply across all timing modes.

Use the **set_budget_options** command to specify general options to be used during creation of timing budgets. To set other specific budget parameters, use the **set_latency_budget_constraints**, and **set_boundary_budget_constraints** commands.

Use the **write_budgets** command to generate SDC that can be applied to lower level blocks. The **compute_budget_constraints** command can be used to set key budget parameters automatically. The **report_budget** command generates useful information about the current budget calculation, allocation, and whether that budget is currently being met. Use the **write_script -include budget** command to save a Tcl script that includes your current settings from this command.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following declares that 30% of each budget path be allocated inside each pin's block:

```
prompt> set_pin_budget_constraints -internal_percent 30 -all
```

For the outputs of the "CPU", allocate 40% inside the block. Note that inputs of the CPU block are not affected:

```
prompt> set_pin_budget_constraints -internal_percent 40 -outputs core/CPU/*
```

For any block input from a "tstclk" clock register, set so the budgeted path delay leading to the block is 2.5ns:

```
prompt> set_pin_budget_constraints -from_delay 2.5 \
  -inputs -from_clock "tstclk" -all
```

For any block output to a "tstclk" clock register, set so no budget is created:

```
prompt> set_pin_budget_constraints -none \
  -outputs -to_clock "tstclk" -all
```

The following example declares a driving cell that is applied to the input pins of "block1". An additional 0.1 of capacitance is also applied to the input:

```
prompt> set_boundary_budget_constraints -name DCbuf8 -driving_cell bufX8
prompt> set_boundary_budget_constraints -name DCbuf8 -default -fanin_capacitance 0.1
prompt> set_pin_budget_constraints -late_boundary DCbuf8 -inputs block1/*
```

The following example sets output loads for all budgeted pins. The value is different for different corners.

```
prompt> set_boundary_budget_constraints -name OUTload -default   -load_capacitance 0.1
prompt> set_boundary_budget_constraints -name OUTload -corner c1 -load_capacitance 0.15
prompt> set_boundary_budget_constraints -name OUTload -corner c2 -load_capacitance 0.05
prompt> set_pin_budget_constraints -late_boundary OUTload -outputs -all
```

Set the same budget for all of the pins of the specified bus:

```
prompt> set_pin_budget_constraints -internal -same_as_pin my_bus[0] \
  -internal U1/U2/my_bus[*]
```

Set the constraints to match the constraints in mode TEST for all pins of block1:

```
prompt> set_pin_budget_constraints -internal -same_as_modes TEST \
  -internal block1/*
```

Set the same budget for the feedthrough of two multiply instantiated instances:

```
prompt> set_pin_budget_constraints -same_as_mib -feedthrough \
  {mib_block1/* mib_block2/*}
```

Set the budgets so that the internal paths of a multiply instantiated instance does not get the same budget as other instances: (By default, internal budgets are the same for multiply instantiate blocks.)

```
prompt> set_pin_budget_constraints -not_same_as_mib -internal mib_block1/*
```

SEE ALSO

compute_budget_constraints(2)
report_budget(2)
set_boundary_budget_constraints(2)
set_budget_options(2)
set_latency_budget_constraints(2)
write_budgets(2)

set_pin_name_synonym

Set pin name synonym for pin object search.

SYNTAX

status **set_pin_name_synonym**
[-full_name]
[-force]
pin_name_synonym *pin_name_synonym*
pin_name *pin_name*

Data Types

pin_name_synonym string *pin_name* string

ARGUMENTS

-full_name

Indicates that a pin name synonym is a full name synonym.

-force

Overwrites the existing pin name synonym if a conflict occurs.

DESCRIPTION

This command defines pin name synonyms, which are supported in pin object queries.

The pin_name_synonym is an alternative pin name for a pin object. For example, if the U1/U2/data_in name is defined as the synonym for the U1/U2/D pin, pin query commands can find the pin object named U1/U2/D when the commands could not find the pin with the given name U1/U2/data_in.

Pin name synonyms are only applicable to pins on leaf-level sequential cells. This feature can also be applicable to physical and logic library pin names.

The tool supports two types of pin name synonyms: full name synonyms and simple name synonyms. The set_pin_name_synonym command defines simple name synonyms unless the -full_name option is used. When defining a full name synonym, you specify the full hierarchical name for the pin object in both the pin_name_synonym and pin_name fields. Pin query commands use full name synonyms exactly as they are defined. Simple name synonym definitions use short names for pin objects. For library pin, only the simple name is needed. Pin query commands use a simple pin name synonym to construct a full pin name during a synonym-based pin object search.

You can get full name and simple name synonyms for pin objects by querying the full_name and name attributes for pin objects.

Wildcards and regular expressions are not supported in pin name synonym definitions. Simple name synonyms do not support the forward slash (/) since it is considered a hierarchical separator.

When you use the pin name synonyms for pin queries, the full name synonym, when applicable, supersedes the simple name synonym. And, the finder commands for pin name synonym can be enabled or disabled by app option of "design.enable_rule_based_query".

EXAMPLES

The following examples use set_pin_name_synonym to define pin name synonyms and show how get_pins command uses the synonyms:

```
prompt> set_pin_name_synonym SYN_CD CD
1

prompt> set_pin_name_synonym -full_name \
      this/is/a/synonym mid1/FJK2SP_at_mid/TI
1

prompt> set_pin_name_synonym -full_name \
      mid1/FJK2SP_at_mid/J mid2/bot2/LSR0P_at_bot/Q
1
```

SEE ALSO

remove_pin_name_synonym(2)
report_pin_name_synonym(2)

set_pipeline_scan_data_configuration

Specifies the pipeline scan data configuration for the design.

SYNTAX

status **set_pipeline_scan_data_configuration**
[-head_pipeline_clock *clock_name*]
[-tail_pipeline_clock *clock_name*]
[-head_pipeline_stages *total_depth*]
[-tail_pipeline_stages *total_depth*]
[-head_shared_pipeline_stages *shared_top_level_depth*]
[-tail_shared_pipeline_stages *shared_top_level_depth*]
[-head_scan_flop true | false]
[-tail_scan_flop true | false]

Data Types

clock_name string
total_depth integer
shared_top_level_depth integer

ARGUMENTS

-head_pipeline_clock *clock_name*

Specifies the shift clock for the head pipeline scan data registers. The specified clock must be previously declared as a scan clock using the **set_dft_signal** command.

-tail_pipeline_clock *clock_name*

Specifies the shift clock for the tail pipeline scan data registers. The specified clock must be previously declared as a scan clock using the **set_dft_signal** command.

-head_pipeline_stages *total_depth*

Specifies the number of head pipeline stages. The default is 1. You can choose not to insert head pipeline stages by specifying a depth of 0.

-tail_pipeline_stages *total_depth*

Specifies the number of tail pipeline stages. The default is 1. You can choose not to insert tail pipeline stages by specifying a depth of 0.

-head_shared_pipeline_stages *shared_top_level_depth*

Specifies the number of top-level shared head pipeline stages to use when shared codec input connections are used for core

integration.

When set, the tool uses the specified number of shared pipeline registers along the shared scan input path, then it uses dedicated pipeline registers as needed for the remaining stages to meet the total head pipeline depth target. A value of 0 uses only dedicated top-level stages. The default is to use the largest number of shared top-level stages possible given the constraints, which minimizes the number of dedicated top-level stages.

This option is only used when shared codec input connections are specified with the **-shared_inputs** option of the **set_scan_compression_configuration** command.

-tail_shared_pipeline_stages ***shared_top_level_depth***

Specifies the number of top-level shared tail pipeline stages to use when shared codec output connections are used for core integration.

When set, the tool uses the specified number of shared pipeline registers along the shared scan output path, then it uses dedicated pipeline registers as needed for the remaining stages to meet the total tail pipeline depth target. A value of 0 uses only dedicated top-level stages. The default is to use the largest number of shared top-level stages possible given the constraints, which minimizes the number of dedicated top-level stages.

This option is used only when shared codec input connections are specified with the **-shared_inputs** and **-shared_outputs** options of the **set_scan_compression_configuration** command.

-head_scan_flop true | **false**

When set to **true**, scan-replaced registers are used for the head pipeline stages. The scan data input of each head pipeline scan register is used for the pipeline shift path. The functional data input of each register is driven by that register's output, so that the register holds state during scan capture. The default is **true**, which inserts scan head pipeline registers.

-tail_scan_flop true | **false**

When set to **true**, scan-replaced registers are used for the tail pipeline stages. The scan data input of each tail pipeline scan register is used for the pipeline shift path. The functional data input of each register is driven by that register's output, so that the register holds state during scan capture. The default is **false**, which inserts nonscan tail pipeline registers.

DESCRIPTION

This command specifies the pipeline scan data configuration for the design.

EXAMPLES

prompt> **set_dft_signal -type MasterClock -port pipeClk -timing [list 45 55]**

prompt> **set_pipeline_scan_data_configuration -head_pipeline_stages 1 -head_pipeline_clock pipeClk**

prompt> **set_pipeline_scan_data_configuration -tail_pipeline_stages 1 -tail_pipeline_clock pipeClk**

SEE ALSO

set_dft_signal(2)
preview_dft(2)

set_pipeline_scan_enable_configuration

Specifies the configuration used for inserting DFT pipeline scan enable logic in the design.

SYNTAX

status **set_pipeline_scan_enable_configuration**
[-exclude *object_list*]
[-exclude_types *type_list*]

Data Types

object_list list
type_list list

ARGUMENTS

-exclude *object_list*

Specifies which elements must be excluded from pipeline scan enable. Those will be connected directly to the relevant scan enable signal instead. The following design elements can be specified as exclusion parameters:

- DFT clock pins: all scan elements clocked by the specified clock pin will be excluded.
- Hierarchies: all scan elements contained by this hierarchy (or any of its subhierarchies) will be excluded.
- Instances or pins: scan enable connections to the specified instance or pin will be excluded.

-exclude_types *type_list*

Specifies which classes of scan elements must be excluded from pipeline scan enable. The available types are:

- shared_wrapper
- dedicated_wrapper
- input_wrapper
- output_wrapper
- input_shared_wrapper
- output_shared_wrapper
- input_dedicated_wrapper
- output_dedicated_wrapper

- testpoint
- clock_gate

DESCRIPTION

The **set_pipeline_scan_enable_configuration** command must be used to specify parameters to further fine-tune insertion of Pipeline Scan Enable (PSE) logic during DFT scan synthesis. However, the master switch to enable this feature is located at **set_scan_configuration -pipeline_scan_enable**.

EXAMPLES

To exclude all elements clocked by a particular clock:

```
> set_dft_signal -type MasterClock -port tclk
> set_pipeline_scan_enable_configuration -exclude tclk
```

The exclusion specification can be a list, and can mix types of elements:

```
> set_pipeline_scan_enable_configuration -exclude { tclk ctl_hier alu/q[1] actl/SE1 }
```

SEE ALSO

set_dft_signal(2)
set_scan_configuration(2)

set_placement_ir_drop_target

Set placer IR drop targets

SYNTAX

string **set_placement_ir_drop_target** [-irdrop]
voltage_area
type
percentage

list *voltage_area*
string *type*
float *percentage*

ARGUMENTS

-irdrop

Specifies that the percentage is a voltage drop number, as a percentage of the effective voltage supply voltage

voltage_area

Specifies the voltage area for settings the IR drop targets

type

Specifies the type of IR drop target setting, either low or high

percentage

Specifies the percentage between 0.0-100.0 of the IR drop target

DESCRIPTION

This command set the IR aware placement target settings for the given voltage area. By default, it set the population percentages. The voltage drop percentages are set, if the option **-irdrop** is used. If the **type** is set to low it will either set the low voltage drop target or the middle bin population, based on the **-irdrop** option is used or not. If the **type** is set to high, it will either set the high voltaged drop target or the upper bin population.

EXAMPLES

The following example sets for the voltage area DEFAULT_VA the low voltage drop target to 4.5% of the effective supply voltage and the high target to 1% of the population

```
prompt> set_placement_ir_drop_target DEFAULT_VA low 4.5 -irdrop
prompt> set_placement_ir_drop_target DEFAULT_VA high 1.0
prompt> get_placement_ir_drop_target DEFAULT_VA
{ { DEFAULT_VA low 4.500000 -irdrop } { DEFAULT_VA high 1.000000 } }
```

SEE ALSO

get_placement_ir_drop_target(2)
reset_placement_ir_drop_target(2)
report_placement_ir_drop_target(2)
place.coarse.ir_drop_default_target_low_population(3)
place.coarse.ir_drop_default_target_low_percentage(3)
place.coarse.ir_drop_default_target_high_population(3)
place.coarse.ir_drop_default_target_high_percentage(3)

set_placement_spacing_label

Specify the placement spacing label that applies to a collection of library cells on left or right side, in R0 orientation, or both.

SYNTAX

status **set_placement_spacing_label**
-name *name*
-side left | right | both
-lib_cells *lib cells*
[-row *row number*]

Data Types

name	string
lib cells	collection of library cells
row number	integer

ARGUMENTS

-name *name*

Specifies a name for this label. This is a required option.

-side *left* | *right* | *both*

Specifies which side, in R0 orientation of the library cell this label applies. Must be one of "*left*", "*right*", or "*both*". This is a required option.

-lib_cells *lib cells*

Specifies the collection of library cells to which to apply this label. This is a required option.

-row *row number*

Specifies row of library cell to apply the label to. The row is in units of the library cell's site_def height. Row number is an integer in range [0..N-1] where N is the number of rows of the library cell. Only applies to multi-height cells. If row number is not provided for multi-height cell, then the label is applied to all the rows of the cell. This is an optional option.

DESCRIPTION

This command assigns labels to a collection of reference (library) cells in order to enable the legalization tool to obey intercell spacing constraints. A label is a string assigned by the user to the left and/or right of a standard cell. This command works in concert

with the command set_placement_spacing_rule that defines the range of spacing between two adjacent labels that is illegal. All spacing rules specified by set_placement_spacing_rule commands need to be satisfied to produce a legal placement. With intercell spacing rule specified, it is no longer assumed that any two standard cells can be placed next to each other with a certain spacing.

The labels and spacing rules can be used to express spacing requirements that originate from mask rules and the layout of standard cells.

Spacing labels and rules are persistent in the NDM.

EXAMPLES

The following example sets labels on a collection of library cells.

```
prompt> set_placement_spacing_label -name X \
  -side both -lib_cells [get_lib_cells -of [get_cells]]
```

The following example sets label on row 1 for a specific multi-height library cell.

```
prompt> set_placement_spacing_label -name X \
  -side left -lib_cells [get_lib_cells -of [get_cell U0]] -row 1
```

SEE ALSO

set_placement_spacing_rule(2)
remove_placement_spacing_rules(2)
report_placement_spacing_rules(2)

set_placement_spacing_rule

Specify the placement spacing rule between library cells that have been assigned labels with set_lib_cell_spacing_label command.

SYNTAX

status **set_placement_spacing_rule**
-labels *label names*
{*min max*}

Data Types

label names	pair of strings
min max	pair of integers

ARGUMENTS

-labels {*label names*}

Lists placement spacing labels. The list must contain exactly two labels specified by the set_placement_spacing_label command. This is a required option.

{*min max*}

Specifies illegal spacing range given in units of sites, where max >= min. This is a required option.

DESCRIPTION

This command assigns inter-cell spacing constraints between reference cells that have been assigned labels with the set_placement_spacing_label command. Cells with the given labels are prohibited from being placed adjacent to one another within the range given by min and max. The range is specified in units of placement sites. Given two adjacent standard cells with an empty space in between, each label from the right side of the left cell is checked against each label from the left side of the right cell to see if any spacing rules are violated. Such pair wise checking of labels is applied to all adjacent cells to ensure placement legality of a chip.

A label typically represents some layout features inside a standard cell. A label is associated with the left or right boundary of a standard cell in the R0 orientation when specified. When a cell is flipped, its associated labels will also be flipped.

All spacing rules specified by set_placement_spacing_rule commands need to be satisfied to produce a legal placement. With placement spacing rule specified, it is no longer assumed that any two standard cells can be placed next to each other with a certain spacing.

The labels and spacing rules can be used to express spacing requirements that originate from mask rules and the layout of

standard cells.

Built-in label 'SNPS_BOUNDARY' may be used to specify spacing rules between library cells and various types of placement boundaries: chip boundary, hard macro edge, hard macro keepout margin, hard placement blockage edge, or voltage area guard bands.

Spacing labels and rules are persistent in the NDM.

EXAMPLES

The following example sets placement spacing rules for labels, X and Y, where spacings of 0, 1, and 2 site units between the cells are illegal.

```
prompt> set_placement_spacing_rule -labels {X Y} {0 2}
```

In the following example, cells with labels "X" are prohibited to be spaced within range 0 to 1 placement sites from placement boundary.

```
prompt> set_placement_spacing_rule -labels {X SNPS_BOUNDARY} {0 1}
```

SEE ALSO

set_placement_spacing_label(2)
remove_placement_spacing_rules(2)
report_placement_spacing_rules(2)

set_placement_status

Sets the placement status of a cell or port.

SYNTAX

status **set_placement_status**
status
object_list

Data Types

status string
object_list collection

ARGUMENTS

status

Specifies the placement status. Valid values are:

- **unplaced**
  The cell or port instance has not yet been placed.
- **placed**
  The cell or port instance has been placed, but can be moved by subsequent operations.
- **legalize_only**
  The cell instance has been placed and can be moved only by the legalizer. This value applies only to cells.
- **fixed**
  The cell or port instance has been placed and the tool cannot move it during subsequent optimization commands, such as **place_opt**. However, you can move the cell or port instance manually, either by using manual editing commands, such as **set_attribute**, or by moving it in the GUI.
- **application_fixed**
  The tool has marked the cell as fixed. This status is similar to the **fixed** status, but the tool can modify this setting and then move the cell. This value applies only to cells.
- **locked**
  The cell or port instance has been placed and cannot be moved either manually or by the tool.

object_list

Specifies the cells or ports for which to set the placement status.

DESCRIPTION

This command sets the placement status of a cell or port.

To query this setting, use the **get_attribute** command to get the value of the **physical_status** attribute for the cell or port.

EXAMPLE

The following example shows how to set the placement status to **fixed** to prevent the placer from moving the cell instance named INST_1.

```
prompt> set_placement_status fixed INST_1
```

The following example shows how to query the placement status.

```
prompt> get_attribute [get_cells INST_1] physical_status
fixed
```

SEE ALSO

set_cell_location(2)
create_placement(2)
get_attribute(2)

set_pocvm_corner_sigma

Selects the standard deviation to be used for parametric on-chip variation analysis when calculating corner values from statistical quantities.

SYNTAX

status **set_pocvm_corner_sigma**
[-corners *corner_list*]
corner_sigma

Data Types

corner_list list
corner_sigma float

ARGUMENTS

-corners *corner_list*

Specifies the corners that the corner sigma value applies to. If this option is not given, the current corner is used.

derate_value

Specifies the corner sigma value.

DESCRIPTION

Parametric on-chip variation (POCV) analysis internally computes arrival times, required times, and slack values based on statistical distributions. When performing comparisons between these statistical quantities, the tool needs to know what values in the distribution are considered worst-case.

The default behavior is to choose values at 3.0 standard deviations away from the mean value. In other words, for an arrival time distribution, the worst-case early arrival is 3.0 standard deviations below the mean, and the worst-case late arrival is 3.0 standard deviations above the mean.

You can use this command set a different number of standard deviations away from the mean to determine the worst-case values for timing reports. Use a lower value such as 2.5 for less worst-case variation and more relaxed timing constraints. Conversely, use a higher value such as 3.5 for more worst-case variation and more restrictive timing constraints.

Note that **set_pocvm_corner_sigma** has higher priority than **time.pocvm_corner_sigma**. When they are both specified, only **set_pocvm_corner_sigma** will be honored.

Multicorner-Multimode Support

By default, this command works on the current corner. To specify a different corner, use the **-corners** option.

EXAMPLES

The following example sets a corner sigma value 3.5 on current corner:

prompt> **set_pocvm_corner_sigma 3.5**

SEE ALSO

time.pocvm_enable_analysis(3)
time.pocvm_corner_sigma(3)

set_pop_up_object_options

Sets options for the **pop_up_objects** command.

SYNTAX

string **set_pop_up_object_options**
-object_type *types*
[-block_action keep | remove]
[-routing_overlap_check true | false]
[-pins_as_terminals true | false]

Data Types

types string

ARGUMENTS

-object_type *types*

Specifies the types of the objects for which to set options. Specify one or more of the supported object types by using the following keywords:

- **pg_routing**
  The **pg_routing** objects include wires, paths, vias, via arrays and terminals that connect to power and ground nets.
- **signal_routing**
  The **signal_routing** objects include wires, paths, vias, via arrays and terminals that connect to signal nets.
- **routing_guide**
  The **routing_guide** objects include all the rectangular route guides.
- **blockage**
  The **blockage** objects include all the placement blockages.
- **pin_guide**
  The **pin_guide** objects include all the rectangular and rectilinear pin guides.
- **pin_blockage**
  The **pin_blockage** objects include all the rectangular and rectilinear pin blockages.
- **cells**
  The **cells** objects include cell instances in child blocks.
- **marker_layer**
  When specifying **marker_layer**, the marker layer shapes in the hard macro cells that are specified in the **pop_up_objects** command are popped up to the top block. When specifying **marker_layer**, you should always specify the **-block_action**

keep.

This is a required option.

-block_action keep | remove

Specifies how to process the objects at the block level. You can specify only one block action type. By default, the block action is remove. If the object_type is **cells**, the only supported block_action is remove.

If you specify **-block_action remove**, the **pop_up_objects** command creates a copy of the object at the top level and deletes the object from the block level. If **-block_action keep** is specified, the **pop_up_objects** command creates a copy of the object at the top level, but keeps the object in the block level. Note that object_type **cells** does not support **-block_action keep**.

-top_action copy | merge

Specifies whether or not to merge the popped up routing path with top level path(s). You can specify only one top action type. By default, the top action is merge.

When **-top_action merge**, then the popped up routing path will be merged with any top level path if 1.they have the same direction; 2.they are logically connected; 3.they can be perfectly merged into one single path. When **-top_action copy**, then the popped up routing path will remain exactly the same as it was in the block.

-routing_overlap_check true | false

Specifies whether overlap checking is performed before popping up the routing objects. This option is valid only for *remove* block action type. The command checks only the overlap of the same type objects. You must use more specific DRC checks to prevent DRC violations and shorts. If overlap exists in the design, the command errors out without popping up the object. By default, the *-routing_overlap_check* option is *false* and the command does not perform overlap checking.

-pins_as_terminals true | false

Specifies whether to create top-level terminals in the parent level, based upon a block pin, when the parent-level net has a top-level port connection. This option can only be used in conjunction with object types pg_routing and signal_routing. A terminal will be created if the child net has been specified for pop-up, or if the block terminal has been specified for pop-up. If a specific object of a child net has been specified in the collection, but not the entire net, and not the specific block terminal, then no terminal will be created.

DESCRIPTION

This command sets options for the **pop_up_objects** command. You can review the options with the **report_pop_up_object_options** command.

EXAMPLES

The following example sets *block_action* to *keep* for *pg_routing* popping up and specifies a routing overlap check before popping up objects.

```
prompt> set_pop_up_object_options -object_type pg_routing \
    -block_action keep \
    -routing_overlap_check true
```

The following example specifies pins to be popped up as top-level terminals if the net is port-connected in the parent level.

```
prompt> set_pop_up_object_options -object_type signal_routing \
        -pins_as_terminals true
```

SEE ALSO

pop_up_objects(2)
remove_pop_up_object_options(2)
report_pop_up_object_options(2)

set_port_antenna_property

Sets antenna attribute values on the specified port.

SYNTAX

collection **set_port_antenna_property**
-port *port*
-data *data*
[-add]
[-replace]

Data Types

port string
data string

ARGUMENTS

-port *port*

Specifies the port (or library cell pin) for which to set the antenna property. This option is required.

-data *data*

Specifies the data used to set the antenna property. This option is required.

-add

Appends additional data to the existing antenna property data for a port or library cell pin. The command replaces only the data that is specified in the command. This mode is optional. If this mode is not specified, **-replace** mode is used.

-replace

Replaces any existing data on the port or library cell pin, even for layers that are not specified in the given data. This mode is optional. By default, **-replace** mode is used.

DESCRIPTION

This command sets antenna property data on the specified port or library cell pin. It returns the data as a Tcl array in the form described in the **get_port_antenna_property** command. If the command is not successful, it returns an empty list.

The format used to specify the antenna data can be in either of the two forms (single- or multiple layer data) as follows;

```
# Single layer data
{
 {[well_type] [oxide_model <type>] metalLayer metalLayer1 gate_size gate_data
  mode1_area mode1_data mode2_ratio mode2_data mode3_area mode3_data
  mode4_area mode4_data mode5_ratio mode5_data mode6_area mode6_data}
}

# Multiple layer data
{
 {{[well_type] [oxide_model <type>] metalLayer metalLayer1 gate_size gate_data1 ...}}
 {{[well_type] [oxide_model <type>] metalLayer metalLayer2 gate_size gate_data1 ...}}
 ...
}
```

For ease of use, you can create a simpler list which contains the data values with the following format:

```
{{[well_type] [oxide_model_type] metalLayer1 gate_data mode1_data mode2_data mode3_data
 mode4_data mode5_data mode6_data}}
```

The data for multiple layers can be set simultaneously by using the following syntax:

```
{
 {{[well_type] [oxide_model_type] metalLayer1 gate_data1 ...}}
 {{[well_type] [oxide_model_type] metalLayer2 gate_data2 ...}}
 ...
}
```

EXAMPLES

The following example sets the antenna property on port p3 for a single layer.

```
prompt> set_port_antenna_property -port p3 \
  -data {{M1 0.3436 6.664 2.05346 12.708 17.5392 7.04044 12.682}}
{metalLayer M1 gate_size 0.3436 mode1_area 6.664
mode2_ratio 2.05346 mode3_area 12.708 mode4_area 17.5392
mode5_ratio 7.04044 mode6_area 12.682}
```

The following example sets the antenna property on port p3 for multiple layers and well dependent.

```
prompt> set_port_antenna_property -port p3 \
   -data {{{M1 0.3436 6.664 2.05346 12.708 17.5392 7.04044 12.682}} \
        {{well2 M2 0.3436 6.664 2.05346 12.708 17.5392 7.04044 12.682}}}
{{metalLayer M1 gate_size 0.3436 mode1_area 6.664
mode2_ratio 2.05346 mode3_area 12.708 mode4_area 17.5392
mode5_ratio 7.04044 mode6_area 12.682} {well2 metalLayer M2 gate_size
0.3436 mode1_area 6.664 mode2_ratio 2.05346 mode3_area 12.708
mode4_area 17.5392 mode5_ratio 7.04044 mode6_area 12.682}}
```

SEE ALSO

get_port_antenna_property(2)

derive_hier_antenna_property(2)

set_port_attributes

Sets the specified attributes and their value on the ports.

SYNTAX

status **set_port_attributes**
[-ports *port_list*]
[-elements *element_list*]
[-exclude_ports *exclude_port_list*]
[-exclude_elements *exclude_element_list*]
[-model *lib_cell_name*]
[-applies_to inputs | outputs | both]
[-attribute *name_value_pair*]
[-clamp_value 0 | 1 | Z | latch]
[-driver_supply *supply_set*]
[-receiver_supply *supply_set*]
[-repeater_supply *supply_set*]
[-feedthrough]
[-unconnected]
[-is_analog]
[-literal_supply *supply_set* | {*supply_net supply_net*}]

Data Types

port_list list
element_list list
exclude_port_list list
exclude_element_list list
supply_set string
direction string
name_value_pair string
supply_set string
supply_net string
model string

ARGUMENTS

-ports *port_list*

Specifies the collection of ports on which the attributes must be set.

-elements *element_list*

Specifies the current scope or list of instance names for whose ports or pins the attributes are applicable.

One of the **-ports** or **-elements** options should be specified for this command.

-exclude_ports *exclude_port_list*

Specifies the collection of ports on which the attributes should not be applied.

-exclude_elements *exclude_element_list*

Specifies the current scope or list of instance names for whose ports or pins the attributes should not be applied .

-model *lib_cell_name*

Specifies the name of lib cell on which the attributes must be set.

-applies_to direction

Specifies whether the given **set_port_attributes** command applies to all input ports or output ports or both input/output ports of the specified element. The argument must be used in conjunction with **-elements** argument. The default value for this argument is **both**. An inout port won't be included with *-applies_to input* or *-applies_to output*.

-attribute *name_value_pair*

Specifies the name and value of the attribute to be set on the ports specified by the **-ports** option.

-clamp_value 0 | 1 | z | latch

Specifies the clamp value to be set on the port.

-driver_supply *supply_set*

Specifies the supply for the logic driving the port.

When an inout port is applied, the value of **-driver_supply** should be electrical equivelant to the value of **-receiver_supply** if specified.

-receiver_supply *supply_set*

Specifies the supply for the logic reading the port.

When an inout port is applied, the value of **-receiver_supply** should be electrical equivelant to the value of **-driver_supply** if specified.

-repeater_supply *supply_set_ref*

Specifies the supply used by the repeater driving the port.

This option is not supported for ports and pins of direction inout.

-feedthrough

Specifies the attribute "feedthrough".

-unconnected

Specifies the attribute "unconnected".

-is_analog

Specifies the attribute "is_analog" to mark the port as an analog port.

-literal_supply ***supply_set*** | **{*supply_net supply_net*}**

Specifies the supply to be used when implementing a literal constant driving the attributed port.

User is expected to give either a supply set reference or a pair of supply net references where the first supply net corresponds to the power supply and the second to the ground supply.

DESCRIPTION

This command sets the attributes and their value on a list of ports specified by the **-ports** option or on a list of instances specified by **-elements** option.

If an attribute is defined with both the **-ports** and **-elements** options, the one specified with the **-ports** option has precedence over the one specified with the **-elements** option regardless of the order in which they are specified.

One of the **-attribute**, **-driver_supply -receiver_supply** or **-repeater_supply** options is required for this command.

Option **-model** and **-element** are mutually exclusive.

Option **-repeater_supply** and **-attribute repeater_power_net|repeater_ground_net** are mutually exclusive.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

1, The following example sets the receiver supply of the port:

prompt> **set_port_attributes -ports {in1} -receiver_supply SS1**

2, The following example sets the receiver supply of the port in1 only, because in2 is in the option exclude_port:

prompt> **set_port_attributes -ports {in1 in2} -exclude_ports {in2} -receiver_supply SS1**

3, The following examples set a literal supply for two different ports using the supply set and supply net pair options

prompt> **set_port_attributes -ports {macro_inst/in1} -literal_supply SS1**

prompt> **set_port_attributes -ports {macro_inst/in2} -literal_supply {VDD VSS}**

SEE ALSO

set_design_attributes(2)
set_related_supply_net(2)

set_power_budget

Sets power budget on different power components for either the current design or a list of cells in the current design.

SYNTAX

status **set_power_budget**
[-scenarios *scenario_list*]
[-exclude *exclude_list*]
[-exclude_groups *group_names*]
[-cell *cell_list*]
[-rails *rail_list*]
budget_value

Data Types

scenario_list list
exclude_list list
group_names list
cell_list list
rail_list list
budget_value float

ARGUMENTS

-scenarios *scenario_list*

Indicates the scenario for which the budget_value is applied. If no scenario is specified the current_scenario is used.

-exclude *cell_list*

Specifies the list of cells which are not to be considered while applying power budget.

-exclude_groups *group_names*

Cells belonging to the specified power groups are not considered while applying power budget.

-cell *cell_list*

Specifies the list of hierarchical instances on which the specified power budget would be applied.

-rails *rail_list*

Specifies a list of supply nets for applying supply-net-specific power budget on the specified hierarchical instances.

budget_value

Specifies the power budget that would be applied on the specified hierarchical instances. Budget value is the expected total power for the specified block/cell in user output unit for power.

DESCRIPTION

Sets power budget on a list of hierarchical instances in the design. The specified power budget and actual computed power for the hierarchical instance would be used to calculate a scaling factor to scale power for all the instance in the hierarchical block, so that scaled power matches the specified power budget for the instance. With power budget set on design, report_power would report scaled power for the design.

Total power budget applied on the hierarchical instance should not be smaller than the calculated actual total leakage power for the hierarchical instance.

Supply-net-specific power budget can also be applied on the hierarchical blocks in which case power associated with the supply net would only be scaled.

Budget can be applied across multiple hierarchy levels - precedence increases with moving down the hierarchy levels in the design. Setting power budget on a hierarchical instance overrides previously set budget, if any.

reset_power_budget command can be used to clear the applied budget(s) on the design.

Applying power budget overwrites the previously specified derate factors. Also, if budget has been applied on the design or on a cell, set_power_derate will not be able to set a derating factor on the design/cell, unless the budget is first cleared using reset_power_budget.

Multicorner-Multimode Support

EXAMPLES

The following example sets a power budget of value 9.5e+04 on hierarchical instance u3/u4 for the scenario SC1.

```
prompt> set_power_budget -scenarios SC1 -cell {u3/u4} 9.5e+04
```

The following example sets a power budget of value 5.2e+07 on the current design.

```
prompt> set_power_budget 5.2e+07
```

The following example sets a power budget of value 2.5e+05 on the rail VDD1 for the hierarchical instance u2/u3.

```
prompt> set_power_budget -cell {u2/u3} -rails {VDD1} 2.5e+05
```

SEE ALSO

get_power_budget(2)
reset_power_budget(2)

report_power_budget(2)
report_power(2)

set_power_clock_scaling

Specifies the clock frequency scaling for power analysis.

SYNTAX

status **set_power_clock_scaling**
[-period *period*]
[-ratio *ratio*]
[-scenarios *scenario_list*]
[*clock_list*]

Data Types

period float
ratio float
scenario_list list
clock_list list

ARGUMENTS

-period *period*

Specify the period of clock which is used in simulation. This option cannot be used if clock obj has not been specified.Period and ratio options cannot be used together.One of period or ratio option needs to be specified.

-ratio *ratio*

Specify the clock period ratio (clock_period_used_in_SAIF / clock_period_used_in_SDC). Period and ratio options cannot be used together.One of period or ratio option needs to be specified.

-scenarios *scenario_list*

Specify the list of scenarios for storing the scaling factors specifed. If not specified, the current scenario is used.

clock_list

Specify a list of clocks in the design. If this option is not used then scenario and ratio needs to be specified.

DESCRIPTION

The clock frequency used in Switching Activity Interchange Format (SAIF) files, from logic simulation for functional verification can differ from the clock frequency used in the Synopsys Design Constraints (SDC) file for timing and power analysis. This command

tells the tool what frequency is used in logic simulation so that that tool can scale averaged power numbers properly.

The *clock_list* argument is a list of clocks in the design that have the specified period or ratio values used in simulation for SAIF generation. Here the ratio is defined as period-in-simulation / period-in-analysis. The command can be used multiple times, once for each clock. If the same clock is used more than once, a warning message will be issued and the clock specified last takes effect. The '-period' and '-ratio' options cannot be used simultaneously. If no clock object is specified, only the ratio option is allowed and the ratio value is applied to all the nets that do not have an associated simulation clock.

The tool only scales the switching activities which are of only SIMULATED type. The command returns 1 on success and 0 on failure.

Note that we will not save the scaled values in the design as they will happen on demand during activity retrieval. For example, if a net has an original toggle rate 1.0 and the -ratio option of the set_power_clock_scaling command is set to 0.1, the scaling results in the net's toggle rate of 0.1 (the original toggle rate 1.0 is not replaced). On retrieving again the value will again be reported as 0.1 and not 0.01.

When period value is specified for a clock, then the scaling ratio is calculated as the ratio of (period of the clock from set_power_clock_scaling cmd) / (period of the fastest clock of an object). This ratio is applied to the retrieved simulated activity of the object. If the ratio value is specified then it is directly applied as the scaling ratio. This ratio is applied to the retrieved simulated activity of the object.

It is to be noted that scaling information is persistent given that the app option 'power.enable_activity_persistency' is set to 'on'.

If the scenario list is empty , the clocks specified will be checked if they have the same mode as the current scenario. If not then the message will be given that the "clock 'clkname' does not have the same mode as the current scenario".If the scenario list is not empty, the clocks specified will be checked if they have the same mode as any of the scenarios. Only if the clock does not have the same mode as any of the scenarios present in the specified list the message will be given that the "clock 'clkname' does not have the same mode as any of the scenarios specified."

EXAMPLES

The following examples describe the behavior of the cmd.

```
prompt> set_power_clock_scaling -period 2.6 clk1
```

The following example sets the ratio for the clock clk1

```
prompt> set_power_clock_scaling -ratio 2.6 clk1
```

The following example sets the ratio for the current scenario.

```
prompt> set_power_clock_scaling -ratio 2.6
```

In the following example the current scenario is sc1 with mode sc1.mode. By default, the clocks have the same mode as the current scenario, unless the mode option is used with get_clocks cmd.

```
prompt> set_power_clock_scaling -ratio 2.6 -scenarios {sc2 default} {clk1}
The clock 'clk1' does not have the same mode as any of the scenarios specified. (POW-045)
```

In the following example the clk1 is fetched for mode sc2.mode which is the mode for the sc2 scenario. Even though clk1 mode is different from the default scenario, the message is not issued as there is one scenario for which the data can be filled.

```
prompt> set_power_clock_scaling -ratio 2.6 -scenarios {sc2 default} \
  [get_clocks -mode sc2.mode {clk1}]
1
```

In the following example there is no scenario specified, but the clk1 is fetched for mode sc2.mode which is the mode for the sc2

scenario. So the message is issued for the current scenario.

```
prompt> set_power_clock_scaling -ratio 2.6 \
  [get_clocks -mode sc2.mode {clk1}]
The clock 'clk1' does not have the same mode as the current scenario. (POW-045)
```

SEE ALSO

get_power_clock_scaling(2)
report_power_clock_scaling(2)
power.enable_activity_persistency(3)

set_power_derate

Sets power derating factors on different power components for either the current design, a list of cells, library cells or all cells in a power group in the current design.

SYNTAX

status **set_power_derate**
[-scenarios *scenario_list*]
[-leakage]
[-switching]
[-internal]
[-groups *group_names*]
derate_value
[*object_list*]

Data Types

scenario_list list
group_names list
derate_value float
object_list list

ARGUMENTS

-scenarios *scenario_list*

Indicates the scenario for which the derate_value is applied. If no scenario is specified the current_scenario is used.

-leakage

Indicates that the derate_value specified should be applied to leakage power only.

-switching

Indicates that the derate_value specified should be applied to switching power only.

-internal

Indicates that the derate_value specified should be applied to internal power only.

-group *group_names*

Specifies the power group to which the derate_value is applied. The derate_value applies directly to each cell object in the power group. Note groups and object_list are mutually exclusive option.

derate_value

Specifies the power derating factor that is applied to the specified power components as a scalar multiplicative factor.

object_list

Specifies current design or a list of cells or library cells to which the specified power derating factor is applied. Note groups and object_list are mutually exclusive option.

DESCRIPTION

Sets power derating factors on different power components for either the current design, a list of cells, or library cells in the current design. If no object or power group is specified, the power derating factor is set on the current design. The power derating factors are used as a scalar multiplicative factor in power calculation.

Power derating factors affect power analysis results in power reports and power attributes. Power analysis results are multiplied by the derating factors. If power derating factors are not specified, the value of 1.0 is assumed.

If the set_power_derate command is used with the -leakage, -internal or -switching, the specified power derating factor is only applied to leakage, internal or switching power accordingly. If none of -leakage, -internal or -switching is specified the power derating factor applies to all power components, which includes internal, switching, and leakage power. First set a generic power derating factor that applies to all power components, and then refine it by setting specific power derating factors for particular power components on particular design objects.

The object_list option can be used to set power specific derating factors on instances (cells or library cells) in the design. On each instance the -switching, -internal, and -leakage options can be used in the same way as previously described to specify exactly how the derating factors should be applied to each instance in the object list.

When applying power derating factors the following priority is used (in decreasing order of precedence):

1) Leaf Cell or Power Group

2) Hierarchical Cell

3) Library Cell

4) Design

When power derating factors are being applied for a cell the tool first checks if they have been set for the cell itself. It then checks if it has been set for any of the cells hierarchical parents. If no derate factors are found, it checks for the library cell and after that it uses the factors set globally on the design.

Use the -group option to set specific derating factors on all the cell objects in particular power groups. Power group can be user generated or default, such as clock_network, memory, register. They are considered as a collection of leaf and hierarchical cells. If a derate factor is set on a power group, the specified derating factors are applied to all cell objects in the power group.

Power derating factors affect power values shown in power reports. The intended usage mode of this command is to first set global or default derates on the design. Then use the object_list option to set specific derate values for cells or library cells in the design. To set derates globally on the design simply issue the command with no object list specified.

Note that if a power budget has been applied on the design or on a specific hierarchical cell by the set_power_budget command, then set_power_derate will not be able to set a derating factor on that design/cell, unless the previously applied budget is first reset using reset_power_budget.

To set derating values back to the default use the reset_power_derate command.

Multicorner-Multimode Support

EXAMPLES

The following example sets a power derating factor of value 0.95 on all cells in the design for the scenario SCN1.

prompt> **set_power_derate -scenario SCN1 0.95**

The following example sets switching power derate and internal power derate to 0.9

prompt> **set_power_derate 0.9 -switching -internal**

The following example sets the power derating factors to 0.5 for all power components on the memory power group

prompt> **set_power_derate 0.5 -groups { memory }**

The following example sets a leakage power derating factor of 1.1 on all instances of library cell IV in the library MY_LIB

prompt> **set_power_derate -leakage 1.1 [get_lib_cells MY_LIB/IV]**

Assuming that cell, "inv" is a particular instantiation of MY_LIB/IV in the design the following command overwrites the power derating factor for the instance "inv" only

prompt> **set_power_derate -leakage 1.05 [get_cells inv]**

SEE ALSO

get_power_derate(2)
reset_power_derate(2)
report_power_derate(2)

set_power_domain_constraints

Sets power domain specific constraints.

SYNTAX

status **set_power_domain_constraints**
[-name *name*]
-object_list *list_of_domains*
[-always_on_strategy single_power | dual_power]
[-disallow_forward_biasing true | false]
[-disallow_reverse_biasing true | false]

Data Types

name string

list_of_domains collection

ARGUMENTS

-object_list *list_of_domains*

Specifies one or more power domains for which the constraints are defined. This is a required option.

-always_on_strategy single_power | dual_power

Specifies the type of always-on cells to be used, either single-power cells or dual-power cells with backup power. The default is **dual_power**.

DESCRIPTION

This command specifies the power domain related constraints.

"-always_on_strategy" specifies the always-on strategy for specified shutdown power domains. The strategy indicates the type of cells to be used for always-on buffering. You can choose to use standard cells with single-power cells, or dual-power cells with a backup supply. This information drives cell selection during optimization.

The dual-power strategy uses special-purpose, always-on cells. Each of these cells has two power pins: a primary pin connected to the primary supply and a backup pin connected to a backup supply net. The cell also has an attribute called always_on which is set to true at the cell level. This type of cell is known as a dual-rail, always-on cell. Specifying dual-power allows optimization to use either single-rail or dual-rail cells.

The single-power strategy uses standard, single-power cells that are placed in dedicated always-on site rows of the power domain's corresponding voltage area. If single-power is specified, only single-rail cells are used by optimization. These always-on sites are powered by the backup power supply. During optimization, the tool does not make any changes to these always-on site rows, so it leaves these cells on the always-on paths. You should create an always-on site row to place the buffer. If you do not create the site rows and the buffering strategy requires it, the tool inserts single-rail cells and issues an MV-058 warning message.

The dual-power strategy does not require always-on site rows and the cells can be placed anywhere in the voltage area.

The "-disallow_forward_biasing" and "-disallow_reverse_biasing" options, can be used respectively, to disallow forward or reverse biasing in all the power domains specified. A cell is forward biased if the supply voltage on a bias PG pin is lower than its related power, or if it is higher than its related ground and reverse biased if the supply voltage on a bias PG pin is higher than its related power, or if it is lower than its related ground. By default, the tool is not constrained and can choose Forward or Reverse biased cells during optimization.

If there are lib-cells specified with map_level_shifter/map_isolation/map_retention strategies on a given domain and the domain also has constraints defined with set_power_domain_constraints; optimization will only pick cells from the map_* strategies. If there are specific insulated-well/forward biased/reverse biased cells that the user wants the tool to pick, these cells must be specified with the -lib_cells options of the map_* strategies.

EXAMPLE

The following example sets the always-on strategy on *switchable_domain* to use single-power standard cells:

```
prompt> set_power_domain_constraints \
        -object_list switchable_domain \
        -always_on_strategy single_power
1
```

SEE ALSO

create_power_domain(2)
create_voltage_area(2)
remove_power_domain(2)
report_power_domain(2)
set_voltage_area(2)

set_power_group

Sets user specified power groups on cells and libcells.

SYNTAX

status **set_power_group**
object_list
-name *group_list*

Data Types

object_list	list
group_list	list

ARGUMENTS

-name *group_list*

Specifies the user specified power group names that needs to be set on specified *object_list*.

object_list

Specifies list of cells and/or libcells for which the *group_list* needs to be set.

DESCRIPTION

This command sets user specified power groups on the specified object list. object list can contain both cell and libcells. If user specified power groups is again set on a cell/libcell which already has user specified power groups, the last set overrides the previous one.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example sets power group 'macro1' on a libcell and cell.

```
prompt> set_power_group -name macro1 {CORE_NPPL_umc13sp/AND2X1M u3/S_reg_0}
```

SEE ALSO

get_power_group(2)
reset_power_group(2)
report_power_groups(2)
get_power_group_objects(2)
report_power(2)

set_power_io_constraints

Sets power constraints related to the placement of I/O driver pads within the specified I/O guides.

SYNTAX

status **set_power_io_constraints**
[-reference_cell *cell_list*]
[-io_guide_object *io_guide_list*]
[-share *bump_share_specification*]
[-ratio *ratio_specification*]
[-spacing *spacing_specification*]
[-offset *offset_specification*]
[*individual_power_constraint_description*]

Data Types

cell_list string
io_guide_list list
bump_share_specification string
ratio_specification string
spacing_specification string
offset_specification string
individual_power_constraint_description string

ARGUMENTS

-reference_cell *cell_list*

Specifies the lib cell names of the power pads. No more than two names can be specified. This option is required.

-io_guide_object *io_guide_list*

Specifies the names or collection of the target I/O guides. If this option is not specified, all I/O guides are selected.

-share *bump_share_specification*

Specifies the maximum number of power pads that can share the same bump. In general, the **place_io** command tries to assign a unique bump to each pad. However, if this option is used with values larger than 1, multiple power pads can be connected to the same bump. The default value is 1.

The *bump_share_specification* accepts the following syntax:

```
-share {maximum_number library_cell}
-share {{maximum_number_1 library_cell_1} {maximum_number_2 library_cell_2}}
```

The maximum number of power pads is specified for each power pad library cell. The value must be a positive integer. The

library_cell value is the corresponding lib cell name. The value-name pair is placed within curly brackets. If two pairs are given, they are placed within a pair of outer curly brackets.

-ratio *ratio_specification*

Specifies the maximum number of signal driver pads that can be placed between two consecutive power pads of the specified type. The option value is given for each lib cell name of the power pads.

The *ratio_specification* accepts the following syntax:

```
-ratio {maximum_number library_cell}
-ratio {{maximum_number_1 library_cell_1} {maximum_number_2 library_cell_2}}
```

The maximum_number value must be a positive integer. The value-name pair is placed in curly brackets. If two pairs are given, they are placed within a pair of outer curly brackets. If this option is not specified, there is no limit to the number of signal pads that can exist between two consecutive power pads of the corresponding type.

Note that in flip-chip designs, signal pads are defined as pads that have at least one signal pad pin with the bump class label (RDL connection). In nonflip-chip designs, signal pads are defined as pads that have at least one signal pad pin.

-spacing *spacing_specification*

Specifies the maximum spacing between consecutive power pads of the corresponding type.

The *spacing_specification* accepts the following syntax:

```
-spacing {maximum_spacing library_cell}
-spacing {{maximum_spacing_1 library_cell_1} {maximum_spacing_2 library_cell_2}}
```

The maximum spacing value is given for each lib cell name of the power pads. The value must be a positive value. The specification format contains a spacing value followed by the corresponding lib cell name. The value-name pair is placed in curly brackets. If two pairs are given, they are placed within a pair of outer curly brackets. If this option is not used, there is no limit on the spacing for two consecutive power pads of the corresponding type.

-offset *offset_specification*

Specifies the maximum distance between the starting point of the I/O guide and the closest edge of the first power pad of the corresponding type.

The *offset_specification* accepts the following syntax:

```
-offset {maximum_offset library_cell}
-offset {{maximum_offset_1 library_cell_1} {maximum_offset_2 library_cell_2}}
```

The offset value is given for each lib cell name of the power pads. The value must be a positive value. The specification format contains a value followed by the corresponding lib cell name. The value-name pair is placed in curly brackets. If two pairs are given, they are placed within a pair of outer curly brackets. If this option is not used, the space can be any value between the starting point of the I/O guide and the closest edge of the first power pad of the corresponding type.

individual_power_constraint_description

Specifies an individual power constraint. This option is mutually exclusive with all options except for *-io_guide_object*. Use this individual power constraints description format to assign multiple power constraints for any I/O guide. The net connectivity of the new power pads can be described. User can also specify the prefix string for all newly created power pads due to the power constraint.

The individual power constraints description accepts the following syntax:

```
{{reference: library_cell_name}
 {ratio: ratio_value}
 {spacing: spacing_value}
```

```
 {offset: offset_value}
 {prefix: prefix_string}
 {connect: {pin_name_1 supply_net_name_1} {pin_name_2 supply_net_name_2} ...}
}
```

The keywords have the following function:

- reference: Specifies the library cell name, similar to the **-reference_cell** option.
- ratio: Specifies the maximum number of signal driver pads that can be placed between two consecutive power pads, similar to the **-ratio** option.
- spacing: Specifies the maximum spacing between consecutive power pads, similar to the **-spacing** option.
- offset: Specifies the maximum distance between the starting point of the I/O guide and the closest edge of the first power pad of the corresponding type, similar to the **-offset** option.
- prefix: Specifies a string that is inserted into the cell instance name.
- connect: Specifies the connection between power pad pins and power nets.

The *connect*: keyword is optional. If you specify this keyword, you must specify a connection for all pins, otherwise the power constraint is not valid and the tool issues a warning message. The pin name is the reference cell pin and the power net name is the net name is a PG net name in the design. The power pads of the same constraint are connected in the same way. If you do not specify a connection for a port, that port of any newly added power pads is unconnected and the tool issues a warning message. In this case, you must connect the unconnected ports using another command.

DESCRIPTION

This command specifies how power driver pads are placed within I/O guides by the **place_io** command, which inserts power pads and places them to satisfy the power constraints specified by this command and the **set_signal_io_constraints** command. The netlist can be changed during I/O placement.

All value-based options are used to set maximum values. As a result, the actual design results can be less than those values. All power I/O constraints are combined with all signal constraints. You are responsible for eliminating any constraint conflicts and avoiding impossible constraints.

Currently, this command can support two styles when specifying constraints. The first style uses the *-reference_cell*, *-ratio*, *-spacing*, and *-offset* command options to describe the all constraints for any I/O guide. With this style, the tool accepts only two types of power driver pads at most and you cannot specify power pad name prefixes or net connectivity. In addition, the latest power constraint setting of an I/O guide overrides all previous settings on the same I/O guide.

The second style specifies individual power constraints using the following format.

```
{{reference: library_cell_name}
 {ratio: ratio_value}
 {spacing: spacing_value}
 {offset: offset_value}
 {prefix: prefix_string}
 {connect: {pin_name_1 supply_net_name_1} {pin_name_2 supply_net_name_2} ...}
}
```

Power constraints are identified based on reference cell and connection. The prefix is not considered. A power constraint is overwritten if both reference cell and connectivity are identical in the separate commands. In this case, the existing values for spacing, ratio, and offset will be overridden. If no power constraint has been defined, the current power constraint is added to the existing power constraint settings on the corresponding I/O guide. As a result, a single I/O guide can have a large number of power

constraints, each with a unique reference cell and pin connectivity specification. The tool issues warning messages if there are more than 10 power constraints for an I/O guide.

Other power pads outside of the constraint set are not considered even if they belong to the same reference type or are named in the same way. It is your responsibility to delete all unused pads before performing power pad synthesis. The prefix specification is used only when new power pads are added into the netlist and is not used to determine which pad can be used for power constraint. If you do not provide a connection specification, pads with different pin connectivity can potentially be used for the power constraint if they belong to the same reference cell. You are responsible to connect these power pads to nets.

EXAMPLES

The following example sets power constraints on two I/O guides named "north" and "south". For power pads of type "VDD", the maximum spacing between any two pads is 650 micron. There can only be 7 signal pads between two consecutive "VDD" pads. The first "VDD" pads cannot be 100 micron away from the starting point of the selected I/O guides. At most 2 "VDD" pads can share the same bump instances.

```
prompt> set_power_io_constraints -io_guide_object {"north" "south"} \
  -reference_cell VDD -offset {100 VDD} -ratio {7 VDD} -share {2 VDD} \
  -spacing {650 VDD}
```

The following example sets power constraints on all I/O guides in the design. There are two types of power pads, with lib cell names as "VDD" and "VSS", respectively. For both types of pad cells, the maximum spacing between power pads of the same type is 650 microns. There can only be 7 signal pads between two consecutive "VDD" pads and 8 signal pads between two consecutive "VSS" pads. There is no offset requirement. Each power pad must have its own bump instance.

```
prompt> set_power_io_constraints \
  -reference_cell {VDD VSS} -ratio {{7 VDD} {8 VSS}} \
  -spacing {{650 VDD} {650 VSS}}
```

Three examples are given below using individual constraints. The three lines are additive because they define different power constraints. The first command defines a constraint for reference cell "libCell1", with a ratio of 6. Any pads inserted will the "VDD" string inserted into the instance name. The "VDD" pin will be connected to the "VDD" supply net, the "VDDIO" pin will be connected to the "VDDIO_R" supply net, and the "VSSIO" pin to the "VSS" supply net. This constraint is restricted to the specified I/O guides. The second command uses the same reference but with a different prefix and different connections, and a different I/O guide. The third is a specification for a different reference cell and applies to all I/O guides.

```
prompt> set_power_io_constraints \
  -io_guide_object {ring1.top ring1.right ring1.bottom} \
  {{reference: libCell1}
   {ratio: 6}
   {prefix: VDD}
   {connect: {VDD VDD} {VDDIO VDDIO_R} {VSSIO VSS}}}

prompt> set_power_io_constraints \
  -io_guide_object {ring1.left} \
  {{reference: libCell_1}
   {ratio: 4}
   {prefix: VDD3}
   {connect: {VDD VDD3} {VDDIO VDDIO_R} {VSSIO VSS}}}

prompt> set_power_io_constraints \
  {{reference: libCell_2}
   {ratio: 6}
   {prefix: VSS}
   {connect: {VDDIO VDDIO_R} {VSS VSS} {VSSIO VSS}}}
```

SEE ALSO

place_io(2)
remove_power_io_constraints(2)
report_power_io_constraints(2)
set_signal_io_constraints(2)

set_power_strategy_attribute

Set power strategy attribute on specified power cells.

SYNTAX

int **set_power_strategy_attribute**
cells
power strategy

Data Types

cells collection or string
power strategy collection of single object

ARGUMENTS

cells

Specifies a collection of the power cells to be associated with specified power strategy.

power strategy

Specifies the power strategy to be associated with specified cells.

DESCRIPTION

This command associates power cells (isolation, level shifter, power switch, and retention cell) with UPF power strategies.

EXAMPLES

The following associates isolation cells with isolation strategies specified.

prompt> **set_power_strategy_attribute iso* [get_power_strategies -domain TOP iso1]**
1

SEE ALSO

set_isolation(2)
set_level_shifter(2)
create_power_switch(2)
set_retention(2)
get_power_strategies(2)

set_power_switch_placement_pattern

Defines a power switch placement pattern.

SYNTAX

status **set_power_switch_placement_pattern**
[-name *pattern_name*]
[-placement_type ring | array]
[-driver *lib_cell*]
[-direction horizontal | vertical]
[-connect_mode hfn | daisy]
[-pattern *pattern_spec*]
[-port_net_name *prefix*]

Data Types

pattern_name string
lib_cell string
pattern_spec string
prefix string

ARGUMENTS

-name *pattern_name*

Specifies the pattern name to use as a handler for the **create_power_switch_array** and **create_power_switch_ring** commands. If a pattern with same name already exists, the pattern definition is updated with the new pattern syntax and settings.

-placement_type ring | array

Specifies the pattern placement type. A pattern can be either *ring* for a pattern or *array* for an array pattern. By default, the value is *array*.

-driver *lib_cell*

Specifies the driver cell for the pattern. *lib_cell* is a library cell name. **-driver** must be specified for an array pattern. This option is ignored when you specify **-placement_type ring**.

-direction horizontal | vertical

Specifies the direction of the pattern. The pattern can be placed either horizontally or vertically. By default, the value is *vertical*. This option is ignored when you specify **-placement_type ring**.

-connect_mode hfn | daisy

Specifies the intra-pattern connection mode. Specify *hfn* to connect the patterns in high fanout mode, or *daisy* to connect the

patterns in daisy chain mode. By default, the value is *hfn*. This option is ignored when you specify **-placement_type ring**.

-pattern *pattern_spec*

Specifies the pattern syntax used in power switch insertion. An example pattern syntax is as follows:

{cell1 {SPACE number} {cell2 n1} {SPACE number} {cell3 n2} }

where

- cell1, cell2 and cell3 are the reference cell names.
- n1 and n2 are integers that specify the number of cell instances to insert. If you do not specify the instance count, the command inserts one instance of the reference cell.
- SPACE is a keyword and must be followed by a number (double or integer) that represents the size of the space in microns. If you do not specify SPACE, the default space is 0.

The following simple pattern example places one instance of cell1, one instance of cell2, and five instances of cell3. The space between cell1 and cell2 is 10 microns and the space between cell2 and cell3 are 0.

-pattern {cell1 {SPACE 10} cell2 {cell3 5})

The **-pattern** option supports nested patterns. The following nested pattern example places one instance of cell1, places three instances of the pattern group {cell2 cell3}, and places one instance of cell1.

-pattern {cell1 {{cell2 cell3} 3} cell1}

-port_net_name *prefix*

Specifies the net name prefix for intra-pattern connection net. If not specified, the default intra-pattern connection net name will be p*, where * is the net number.

DESCRIPTION

This command specifies the power switch placement pattern and the placement options. If the placement pattern is already defined, the tool updates the definition with the new options. Placement pattern settings are not persistent in the database.

EXAMPLES

The following example defines an array pattern named pattern1 with a vertical direction and HFN connection mode. The driver switch is HEADX2_LVT.

```
prompt> set_power_switch_placement_pattern -name pattern1 \
  -placement_type array -direction vertical -connect_mode hfn \
  -driver HEADX2_LVT \
  -pattern {HEADX2_LVT {SPACE 10} {{HEADX4_LVT {SPACE 8}} 3} }
```

The following example defines a ring pattern named pattern2.

```
prompt> set_power_switch_placement_pattern -name pattern2 \
  -placement_type ring \
  -pattern {HEADX2_LVT {SPACE 10} {{HEADX4_LVT {SPACE 8}} 3} }
```

SEE ALSO

create_power_switch_ring(2)
create_power_switch_array(2)
report_power_switch_placement_patterns(2)

set_process_label

Sets the process label for one or more corners, for part or all of the current design, or only for cells from specific reference libraries.

SYNTAX

string **set_process_label**
[*label*]
[-early *label*]
[-late *label*]
[-clear]
[-corners *corner_list*]
[-object_list *object_list*]
[-library *library*]

list *corner_list*
list *object_list*
object *library*
string *label*

ARGUMENTS

label

The process label to be used for both early and late analysis. If this option is given, the **-early** and **-late** options cannot be given. A value of "*" represents a wildcard: any available process label will be acceptable. In this case, the **set_process_number** command should be used to specify the desired process condition.

-early *label*

The early process label.

-late *label*

The late process label.

-clear

If this option is given, any early/late process label settings on the given objects and/or libs will be removed.

-corners *corner_list*

The corner(s) to apply the process label to. If this option is not given, the current corner is used.

-object_list *object_list*

The cell or port object(s) to apply the process label to. If this option is not given, the process label is applied to the entire current design.

-library ***library***

The reference library to apply the process label to. If this option is not given, the process label is applied to cells from all reference libraries.

DESCRIPTION

The **set_process_label** command is used set the process label parameters that will be used for timing analysis and delay calculation. If no **-object_list** option is given, the given parameters will used for the entire design. If the **-object_list** option specifies a hierarchical block, the parameters will be applied to all leaf cells within that block. If the **-library** option is given, the parameters will be applied only to cells from that library.

It is often necessary to specify one process label for the main cell libraries, and a different label for cells (such as memory or IP blocks) from a specialized library. In the past this was accomplished by giving a separate **set_process_label**, or **set_operating_conditions** command for every cell from the specialized library. When the **-library** option is used, it is only necessary to give a single command, and it will automatically be applied to all cells from that library.

Note that when you set a process label for a design object with **set_process_label**, the tool will look for timing information in a similarly labeled reference library that was created during library preparation. During library preparation, process labels are associated with reference libraries by using the **read_lib**, **read_db**, or **set_process** command.

Multicorner-Multimode Support

By default, this command works on the current corner. To specify a different corner, use the **-corners** option.

EXAMPLES

This example shows how to set the early and late process labels to the same values for an entire design.

```
prompt> set_process_label WORST
```

This example shows how to set the early and late process labels to different values for an entire design.

```
prompt> set_process_label -late WORST -early BEST
```

This example shows how to give default process labels for the design, as well as different process labels for cells from a particular reference library.

```
prompt> set_process_label -late WORST -early BEST
prompt> set_process_label -library mem_lib -late SLOW -early FAST
```

This example shows how to give default process labels for the design, but have process labels be ignored for cells from a particular reference library.

```
prompt> set_process_label -late WORST -early BEST
```

```
prompt> set_process_label -library pll_lib *
```

SEE ALSO

read_db(2)
read_lib(2)
report_pvt(2)
set_process(2)
set_process_number(2)
set_temperature(2)
set_voltage(2)

set_process_number

Sets the process number for one or more corners, for part or all of the current design, or only for cells from specific reference libraries.

SYNTAX

status **set_process_number**
[*number*]
[-early *number*]
[-late *number*]
[-clear]
[-corners *corner_list*]
[-object_list *object_list*]
[-library *library*]

Data Types

number float
corner_list list
object_list list
library string

ARGUMENTS

number

Specifies the process number to be used for both early and late analysis.

If you specify this option, you cannot specify the **-early** and **-late** options.

-early *number*

Specifies the early process number.

If you specify option, you must also specify the **-late** option.

-late *number*

Specifies the late process number.

If you specify option, you must also specify the **-early** option.

-clear

Removes any early or late process label settings on the specified objects and libraries.

-corners ***corner_list***

Specifies the corners to which to apply the process number.

If you do not specify this option, the tool applies the process number to the current corner.

-object_list ***object_list***

Specifies the cells or ports to which to apply the process number. If you specify a hierarchical block, the tool applies the process number to all leaf cells within that block.

If you do not specify this option, the tool applies the process number to the entire current design.

-library ***library***

Specifies the reference library to which to apply the process number. The tool applies the process number to all cells in the specified library.

You can use this option to specify one process number for the main cell libraries and a different number for cells such as memory or IP blocks from a specialized library.

If you do not specify this option, the tool applies the process number to the entire current design.

DESCRIPTION

The **set_process_number** command sets the process numbers that are used for timing analysis and delay calculation.

This command should not be confused with the **set_process** command, which exists only in the library manager, and is used to define the process numbers of a reference library being built.

Multicorner-Multimode Support

By default, this command works on the current corner. To specify a different corner, use the **-corners** option.

EXAMPLES

This example sets the same early and late process number for an entire design.

```
prompt> set_process_number 1.0
```

This example sets separate early and late process numbers for an entire design.

```
prompt> set_process_number -late 1.5 -early 0.8
```

This example sets default process numbers for the design, as well as different process numbers for cells from a particular reference library.

```
prompt> set_process_number -late 1.5 -early 0.8
prompt> set_process_number -library mem_lib 1.0
```

SEE ALSO

set_process_label(2)
set_voltage(2)
set_temperature(2)
report_pvt(2)

set_programmable_spare_cell_mapping_rule

Creates a mapping rule for programmable spare cells (PSC). The rule is used during spare cell mapping with the **place_freeze_silicon** command.

SYNTAX

status **set_programmable_spare_cell_mapping_rule**
-psc_type_id *id_list* | -compatible *id_list*
[-dont_overlap_pg_stripe_layer *layers*]
[-partial_overlap_pg_stripe_layer *layers*]
[-multi_height_split *id_list*]
[-multi_height_merge *id_list*]

Data Types

id_list list
layers collection

ARGUMENTS

-psc_type_id *id_list*

Specifies the list of target psc_type_id for which to set the mapping rule. The list can have one or more ids. This option is mutually exclusive with the **-compatible** option.

-compatible *id_list*

Specifies a list of psc_type_ids which are compatible and must have the same site height. PSC cells of these types can be derived from each other. PSC cells can be used to map ECO cells of compatible types. Note that the PSC mapping between these compatible ids should also honor the other mapping rules set on these type, such as the **-dont_overlap_pg_stripe_layer**. The PSC mapping should consider the PG rules without any PG rule violation. This option is mutually exclusive with the **-psc_type_id** option.

-dont_overlap_pg_stripe_layer *layers*

Specifies the layers of PG stripes that cannot overlap target PSC types. This option must be used together with the **-psc_type_id** option. This option is mutually exclusive with the **-partial_overlap_pg_stripe_layer**, **-multi_height_split**, and **-multi_height_merge** options.

-partial_overlap_pg_stripe_layer *layers*

Specifies the layers of PG stripes and cells of the target PSC types that can partially overlap PG stripes. The allowed overlap area is the right two sites of the cell. This option must be used together with the **-psc_type_id** option. This option is mutually exclusive with the **-dont_overlap_pg_stripe_layer**, **-multi_height_split** and **-multi_height_merge** options.

-multi_height_split *id_list*

Specifies a list contain one psc_type_id which can be derived by splitting from the current PSC types specified by **-psc_type_id**. The multiheight PSC cells of the current PSC types can be split to several small site height PSC cells of the types specified in this option. Especially the PSC types in this list must have different site height. This option must be used together with the **-psc_type_id** option.

-multi_height_merge *id_list*

Specifies a list of psc_type_ids which can be derived by merging the current PSC types specified by **-psc_type_id**. The multiheight PSC cells of the current PSC types can be merged from the small site height PSC cells of the types specified in this option. This option must be used together with the **-psc_type_id** option.

DESCRIPTION

This command sets rules for specified psc_type_ids used during PSC mapping with the **place_freeze_silicon** command. The PSC mapping engine considers these rules during mapping.

You can review the current mapping rules with the **report_programmable_spare_cell_mapping_rule** command. You can remove mapping rules with the **remove_programmable_spare_cell_mapping_rule** command. To ignore all mapping rules, remove all mapping rules before running the **place_freeze_silicon** command.

The currently supported mapping rules are as follows:

1. Don't overlap with PG stripes of the specified layers.

When setting this rule for a psc_type_id, ECO cells of this psc_type_id are not allowed to overlap PG stripes of the specified layers. This means that any part of the cell is not allowed to overlap with these stripes.

2. Partial overlap with PG stripes of the specified layers.

When setting this rule for a psc_type_id, ECO cells of this psc_type_id can only partially overlap PG stripes of the specified layers on the right two sites of the cell. Other areas of the cell are not allowed to overlap with these stripes.

3. Multiple height PSC cells can be split to small site height cells.

When setting this rule for a psc_type_id, PSC cells of this psc_type_id can be derived by splitting to small site height PSC cells of the psc_type_id specified by the **-multi_height_split** option.

4. Multiple height PSC cells can be merged from small site height cells.

When setting this rule for a psc_type_id, PSC cells of this psc_type_id can be derived by merging from small site height PSC cells of the psc_type_id specified in the option **-multi_height_split**.

5. PSC types are compatible

When setting this rule for list of psc_type_ids, PSC cells of these types can be derived from each other.

For rules 3, 4 and 5, the newly derived PSC cells can be used to map ECO cells of the corresponding types. Note that when using newly derived PSC cells from other types by merging, split or compatible conversion, the PSC mapping should also honor PG overlap rules specified by **-dont_overlap_pg_stripe_layer** and **-partial_overlap_pg_stripe_layer** and so on. The PSC mapping should consider the PG rules without any PG rule violation.

When setting rules such as the compatible rule and split/merge rule for deriving new PSC cells, each PSC type should have the same PG stripe layer setting or have no PG rule. Otherwise, the rule setting command reports an error and does not create the rule. It is recommended that you set overlap PG rules before the compatible and split/merge rules to avoid errors during rule setting.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example specifies that ECO cells with type 1 are not allowed to be overlapped with PG stripes of M1 layers.

```
prompt> set_programmable_spare_cell_mapping_rule \
  -psc_type_id 1 -dont_overlap_pg_stripe_layer {M1}
```

The following example specifies that ECO cells with psc_type_id 2 or 3 are only allowed to partially overlap on the right side with PG stripes of M1 layers.

```
prompt> set_programmable_spare_cell_mapping_rule -psc_type_id {2 3} \
  -partial_overlap_pg_stripe_layer {M1}
```

The following example specifies that ECO cells with psc_type_id 3 are multiple site height (3 site height) and can be split to small site height cells with type 1 or 2 (type 1 is single site height, type 2 is double site height).

```
prompt> set_programmable_spare_cell_mapping_rule \
  -psc_type_id 3 -multi_height_split {1 2}
```

The following example specifies that ECO cells with psc_type_id 3 are multiple site height (3 site height) and can be merged from small site height cells with type 1 or 2 (type 1 is single site height, type 2 is double site height).

```
prompt> set_programmable_spare_cell_mapping_rule \
  -psc_type_id 3 -multi_height_merge {1 2}
```

The following example specifies that ECO cells with type 1 and 2 are compatible and have the same site height. ECO cell with type 1 can be mapped to PSC cell with type 2 and the ECO cell with type 2 also can be mapped to PSC cell with type 1.

```
prompt> set_programmable_spare_cell_mapping_rule -compatible {1 2}
```

SEE ALSO

check_freeze_silicon(2)
map_freeze_silicon(2)
place_freeze_silicon(2)
remove_programmable_spare_cell_mapping_rule(2)
report_programmable_spare_cell_mapping_rule(2)

set_propagated_clock

Specifies propagated clock latency.

SYNTAX

string **set_propagated_clock** *object_list*

list *object_list*

ARGUMENTS

object_list

Specifies a list of clocks, ports, or pins.

DESCRIPTION

Specifies that delays be propagated through the clock network to determine latency at register clock pins. If not specified, ideal clocking is assumed. Ideal clocking means clock networks have a specified latency (designated by the **set_clock_latency** command), or zero latency, by default. Propagated clock latency is used for post-layout, after final clock tree generation. Ideal clock latency provides an estimate of the clock tree for pre-layout.

If the **set_propagated_clock** command is applied to pins or ports, it affects all register clock pins in the transitive fanout of the pins or ports. If it is applied to an object (clock, port, or pin) on which network latency is already defined, then a warning is issued.

Virtual clocks do not have any source, and they cannot affect any register in the design. Thus, virtual clocks cannot be made propagated, and a warning is issued if set_propagated_clock is applied on a virtual clock.

To undo **set_propagated_clock**, use the **remove_propagated_clock** command or use the **set_clock_latency** command to provide an ideal latency.

To see propagated clock attributes on clocks, use the **report_clock** command.

Multicorner-Multimode Support

This command works on the current mode.

EXAMPLES

The following example specifies to use propagated clock latency for all clocks only in the current mode in the design.

```
prompt> set_propagated_clock [all_clocks]
```

The following example specifies to use propagated clock latency for all real clocks in the design.

```
prompt> foreach_in_collection mode [all_modes] {
      set_propagated_clock [get_clocks -mode $mode -filter defined(sources)]
    }
```

WARNING: After clock tree synthesis, if we explicitly make all clocks propagated, it can cause I/O clock latencies to become zero. This does not happen if we have either used independent virtual clocks for the I/Os or when the clock network latencies are included in the I/O data path delay. Virtual clocks are not propagated.

SEE ALSO

timing_remove_clock_reconvergence_pessimism(3)
remove_propagated_clock(2)
report_clock(2)
set_clock_latency(2)

set_pt_options

Sets configuration options to optimize the current design with PrimeTime ECO.

SYNTAX

status **set_pt_options**
[-pt_exec *path_name*]
[-host_option *option_name*]
[-scenario_scripts *pair_list*]
[-scenario_constraints *pair_list*]
[-pre_dmsa_script *file_name*]
[-pre_link_script *file_name*]
[-link_script *file_name*]
[-post_link_script *file_name*]
[-work_dir *directory_name*]
[-hold_fix_buffers *buffer_list*]
[-verilog_options *options*]
[-reset]

Data Types

path_name string
option_name string
pair_list list
file_name string
directory_name string
buffer_list list

ARGUMENTS

-pt_exec *path_name*

Specifies the path to the 'pt_shell' PrimeTime executable.

-host_option *option_name*

Specifies the option name defined by the **set_host_options** command. After you specify host options like the number of cores and submit command, PrimeTime slave process will be launched based on the specified options. But **set_host_options -target PrimeTime** has higher priority than **-host_option** for launching PrimeTime slave process. In hierarchical ECO fusion, this host option is passed to the **run_block_script** command in the background to export data for blocks or implement ECO changes using distributed processing. This option is mandatory for hierarchical ECO fusion. After setting the host options, use the **check_host_options** command to validate the settings.

-scenario_scripts *pair_list*

Specifies a signoff script for each signoff scenario using the following format:

{{scen1 script1} ... {scenN scriptN}}

By default, eco fusion will write out verilog, upf, parastics and saif if applicable and define **eco_fusion_verilog**, **eco_fusion_upf**, **eco_fusion_parastic**, **eco_fusion_parasitic_corner_name** and **eco_fusion_read_saif_callback** TCL variables or call back functions to point to corresponding files in the top-level wrapper script assembled for each scenario. These TCL variables could be referred to in end-user provided signoff scripts, which are sourced at end of the top-level wrapper script. But if either number of scenarios or scenario names does not match ICC scenarios exactly, it will only provide pointer to verilog and upf while not parasitic and saif. Then end-users need to prepare the saif/parasitics on their own.

To avoid checking out PrimePower license in eco fusion, eco fusion defines **eco_fusion_read_saif_callback** and calls it automatically from the master process. so end-users should not read saif or call **eco_fusion_read_saif_callback** directly in provided signoff script. But end-user could refine **eco_fusion_read_saif_callback** in provided signoff scripts to read saif file prepared by themselves.

As for hierarchical eco fusion flow, end-users should call **eco_fusion_read_parasitic_callback** to read parasitics for top-block and sub blocks as eco_fusion_parasitic only points to top-block parasitic.

-scenario_constraints ***pair_list***

Specifies a signoff constraint file for each active scenario using the following format:

{{scenario_1 /path_to_file_1} {scenario_2 /path_to_file_2} ...}

where the path is an absolute path to the file. This option is mandatory for the hierarchical ECO fusion flow when a complete full-chip constraint is not available in memory for the **eco_opt** command to export. This can occur when one or more blocks are linked to abstract view instead of the design view.

-pre_dmsa_script ***file_name***

Specifies the file that contains a custom PrimeTime script, which is executed on master before dmsa is launched.

-pre_link_script ***file_name***

Specifies the file that contains a custom PrimeTime script, which is executed on slave before linking in PrimeTime. Use PrimeTime commands that do not require the current design.

-link_script ***file_name***

Specifies the file that contains a custom PrimeTime script, which is executed on slave immediately after linking in PrimeTime. Use PrimeTime commands that require the current design.

-post_link_script ***file_name***

Specifies the file that contains a custom PrimeTime script, which is executed on slave after linking but before update timing in PrimeTime.

-work_dir ***directory_name***

Specifies the directory where PrimeTime files and logs are generated. If you don't specify a working directory, a directory is automatically created under your current working directory.

-hold_fix_buffers ***list***

Specifies a list of library buffer names for hold time fixing.

-verilog_options ***options***

Specifies custom options for write_verilog.

-reset

Resets all command options.

DESCRIPTION

This command specifies options to run PrimeTime from within an implementation tool. For example, you must specify a path to the PrimeTime executable to run PrimeTime ECO.

You can reset to default values for all options by specifying the **-reset** option.

The tools are configured as in the **analyze_timing_correlation** command. If you want to have a different setting for PrimeTime, use the **-pre_link_script** or **-post_link_script** to specify your own PrimeTime setting. The setting will be applied across all scenarios.

For example, if you have turned off crosstalk analysis but want PrimeTime to run in crosstalk analysis mode, create a Tcl script that contains the following PrimeTime command.

```
set_app si_enable_analysis true
```

Suppose the file name is my_sta_setting.tcl. With the following command, PrimeTime will run in SI mode.

```
prompt> set_pt_options -pre_link_script my_sta_setting.tcl
```

To create a setting that requires the current design, you can use the **-post_link_script** option as follows:

```
prompt> set_pt_options -post_link_script my_post_link_setting.tcl
```

If you have multiple scenarios in your design, PrimeTime runs Distributed Multi Scenario Analysis (DMSA) with separate slave processes. If you don't specify any submit commands, the tool launches PrimeTime master and slave processes on the same host machine as the current session. If you specify a submit command, PrimeTime slaves will be launched using the submit command. To specify a submit command, first use the **set_host_options** command to define your host option name. Next, use this option name to specify the **-host_option** option of the **set_pt_options** command. The following example shows the commands using the bsub command to use LSF farm.

```
prompt> set_host_option -name pteco_host_option -submit_command "/lsf/bin/bsub -R \"rusage\[mem=4000\]\""
prompt> set_pt_options -host_option "pteco_host_option"
prompt> report_pt_options
```

EXAMPLES

The following example sets a specific PrimeTime executable.

```
prompt> set_pt_options -pt_exec /u/mgr/bin/pt_shell
```

The following example sets a specific PrimeTime executable and enable In-Design StarRC for extraction.

```
prompt> set_pt_options -pt_exec /u/mgr/bin/pt_shell
```

SEE ALSO

check_pt_qor(2)

eco_opt(2)
report_pt_options(2)

set_push_down_object_options

Sets options for the **push_down_objects** command to push down objects.

SYNTAX

string **set_push_down_object_options**
-object_type *types*
[-top_action keep | remove]
[-block_action {"values are object-type dependent"}]
[-routing_overlap_check true | false]
[-ignore_misalignment true | false]
[-allow_multi_rail_cells true | false]
[-location_based_terminal_naming true | false]
[-collinear_margin *distance*]
[-pin_meet_fatwire_rule true | false]

Data Types

types list
distance float

ARGUMENTS

-object_type *types*

Specifies the types of the objects for which to set options. Specify one or more of the supported object types by using the following keywords:

- **pg_routing**
  The **pg_routing** objects include wires, paths, vias, via arrays and terminals that connect to power and ground nets. Note that this object type is also used to specify push-down options for routing shapes that have no net association.

  The block_actions available for pg_routing push-down are copy, create_blockage, center_pin_from_wire, and create_pin_shape. **copy** and **create_blockage** and **center_pin_from_wire** are mutually exclusive actions. **create_pin_shape** can be used in combination with either of the other 2 block actions. When **copy** is specified, a real copy of the routing (path, via, rectangle) is created in the block. But if **create_blockage** is specified, then a routing blockage is created with the same dimensions as the original routing object. The blockage has no net association, so there is no change to the netlist when routing is pushed down in the form of blockages. And the original routing objects are retained in the top cell. If the **center_pin_from_wire** action is specified, the command will create a center pin from a wire that is in the center of the block. Use caution with this action as it creates a center pin for any wire in the interior of the block; be sure to use this option only with a limited number of wires. And be sure to turn this option back off for normal push-down behavior. The default block_action for pg_routing is **create_pin_shape**.

  The **-collinear_margin** and **-pin_meet_fatwire_rule** options are DRC-avoidance options and are exclusive to power and ground (PG) routing push-down.

The **-collinear_margin** option specifies a distance within which PG routes parallel to (but don't overlap) the block boundary are considered collinear PG routes. A normal collinear PG strap is one that runs parallel to an edge, but overlaps it so that one edge is inside the boundary and the other edge is outside the boundary. Normally, PG straps which run parallel, but do not overlap the boundary, are not treated as collinear straps. Collinear straps are only created in the child block as copies of the original top-level strap, and the original strap is left untouched in the top. With this option, the command will include non-overlapping PG straps and include them as copies in the child block so that their DRC effect can be observed in the block. These will appear outside of the block boundary in the child reference block.

Vias are also processed with this option. However, vias don't typically have a vertical or horizontal direction, and are included by the tool whenever they fall within the extended collinear margin. They are copied into a block with the "copied-down" shadow status, and the original source via in the top is left there.

By default, this option is -1, meaning that the command will only consider overlapping parallel PG straps as collinear straps, and overlapping vias as collinear vias.

If the value of the **-collinear_margin** option is 0 (as in microns), only non-overlapping parallel straps which exactly abut a boundary edge will be pushed into the block as a collinear strap. If the value is greater than zero, any PG strap running parallel to a block edge will be copied into the block as a collinear strap if the closest edge of the strap is within the distance specified in the option.

The **-pin_meet_fatwire_rule** Boolean option is false by default. If this option is true, the command ensures that the size of the PG pin is big enough to trigger fat-wire spacing rules within the block. To do this, the command creates a pin that is as long (in the direction perpendicular to the edge) as it is wide. This length is created inside the boundary if it can safely be done. But if the strap which was the basis for the pin only overlapped the block less than that length, then the pin can protrude outside the boundary.

- **signal_routing**
  The **signal_routing** objects include wires, paths, vias, via arrays and terminals that connect to signal and clock nets. Note that the options set for **signal_routing** apply only to nets that are specifically included in the object collection passed into the **push_down_objects** command. This option does not affect routes between cells that are pushed down; routes between cells and routing associated with internal nets are automatically pushed down when the cells are pushed down. Please note also that this object type is not used to specify push-down options for routing shapes that have no net association. For such non-net shapes, set options by using the pg_routing object type.

  The block_actions available for signal_routing push-down are copy, create_blockage, center_pin_from_wire, and create_pin_shape. **copy** and **create_blockage** are mutually exclusive actions. **create_pin_shape** can be used in combination with either of the other two block actions. When **copy** is specified, a real copy of the routing (path, via, rectangle) is created in the block. But if **create_blockage** is specified, then a routing blockage is created with the same dimensions as the original routing object. The blockage has no net association, so there is no change to the netlist when routing is pushed down in the form of blockages. And the original routing objects are retained in the top cell. If the **center_pin_from_wire** action is specified, the command will create a center pin from a wire that is in the center of the block. Use caution with this action as it creates a center pin for any wire in the interior of the block; be sure to use this option in only with a limited number of wires. And be sure to turn this option back off for normal push-down behavior. The default block_action for signal_routing is **create_pin_shape**.

- **routing_guide**
  The **routing_guide** objects include all the rectangular route guides.

- **routing_corridor**
  The **routing_corridor** objects include all routing corridors. The top_actions available are restricted to just one, "keep", which is the default value. And the only block action available is "copy" (also the default, meaning that the copied-down object will be a routing corridor like the original).

- **blockage**
  The **blockage** objects include all the placement, shaping, and routing blockages.

- **pin_guide**
  The **pin_guide** objects include all the rectangular and rectilinear pin guides.

- **pin_blockage**
  The **pin_blockage** objects include all the rectangular and rectilinear pin blockages.

- **cells**
  The **cells** object_type is used for specifying options related to pushing down cell instances into blocks. The only supported top and block actions for **cells** are **-top_action remove** and **-top_action keep** (just for physical-only cells), and **-block_action: copy | {create_blockage create_route_blockage_from_cell} and retain_obsolete_ports**. The default top action is remove and the default block action is copy. A mutually exclusive action to copy is {create_blockage create_route_blockage_from_cell}. The block_actions create_blockage and create_route_blockage_from_cell are both mutually exclusive to copy, and can be used together or alone, as long as it is not used in combination with copy.

  The create_blockage action will cause the **push_down_objects** command to create a placement blockage in the block rather than an actual cell. The create_route_blockage_from_cell action will similarly cause the **push_down_objects** command to create a routing blockage in the block rather than an actual cell. In both of these blockage cases, there is no change to the netlist in the top block or in the push-down block. When a placement blockage is created in the block as a result of using the create_blockage block action, the new placement or routing blockage is given a shadow status of "copied-down". Such blockages cannot be popped back up, as no objects are eligible for pop-up if they have the "copied-down" shadow status. The block_action for cells, retain_obsolete_ports, will direct the **push_down_objects** command to suppress the deletion of ports that have become obsolete as the result of pushing down cells. This usually occurs when a cell being pushed down has an existing connection to the block, but does not connect to anything else, so that when the cell is pushed down, the port becomes obsolete and the net becomes an internal net in the block.

  If the top_action is "keep", the command only processes physical-only cells in that push-down session, and disregards any non-physical-only cells passed into the command. The physical-only cells are created in the destination block with the "copied-down" shadow status, and the physical-only cells in the top are left in the top. No port changes are made on the block, and no nets are created or modified.

  The **cells** object type includes an option, **-allow_multi_rail_cells true | false** option described below.

- **charging_station**
  The **charging_station** objects include all non-primary voltage areas. The only available top_action is "remove", which is the default value. And the only block action available is "copy".

- **marker_layer**
  When specifying **marker_layer**, the marker layer shapes which are provided to **push_down_objects** command will be pushed down into the overlapped blocks. When specifying **marker_layer**, user should always specify the block action to be "copy".

This option is required.

-top_action keep | remove

Specifies how to process the objects at the top level. You can specify only one top action type. The detailed descriptions for top action types for each object type is described in the **push_down_objects** command man page.

-block_action *{values are object_type dependent}*

Specifies how to process the objects at the block level. You can specify one or more supported block action types. The detailed descriptions of block action types for each object type is described in the **push_down_objects** command man page.

-routing_overlap_check true | false

Specifies whether overlap checking is performed before pushing down the routing objects. This option is valid only when you specify **-block_action copy**. By default, this option is false and no overlap checking is performed before pushing down routing objects.

-ignore_misalignment true | false

Specifies whether the *push_down_objects* command pushes down objects if there are misaligned objects among multiple instantiated blocks (MIBs). If this option is *true*, the *push_down_objects* command issues a warning message and continues

pushing down objects into blocks, even if routing or site rows are misaligned in an MIB design. This option can only be used in combination with object_type pg_routing. By default, this option is false and the **push_down_objects** command does not push down objects if a misalignment exists.

-allow_multi_rail_cells true | false

Specifies whether the *push_down_objects* command pushes down multivoltage (MV) cells with multi-rails if the cell has no associated MV strategy. Note that **-allow_multi_rail_cells true** should be used with caution; the command does not update MV constraints, nets, or ports associated with the cell. It is your responsibility to update the UPF.

This option can only be used in combination with object_type **cells**. By default, this option is false and the **push_down_objects** command does not push down multi-rail UPF cells.

-location_based_terminal_naming true | false

Specifies whether the *push_down_objects* command creates pins with names that are location- and layer-based. By default, the *push_down_objects* command always creates the first pin for a particular port by using the same name as the port; if subsequent pins are created again for that same port, the name is uniquified by appending a incremented number for each additional pin for that port. But this default naming scheme does not ensure that a particular pin at a particular location and layer will always have the same incremented counter number, so the name is not guaranteed to be persistent across multiple runs of this command. So this option allows you to specify that an absolutely unique name will be used for a given location and layer of a pin created for a port. The location is the center point of the pin in the block coordinate system, and the layer is the layer that the pin is created in. This is primarily used in cases where multiple pins shapes can be created for the same port on the block (EEQ pins). This option can only be used in combination with object_type pg_routing or signal_routing. By default, this option is false. Note that this terminal naming option will only be observed in the push-down of routing for nets that have NOT been specified to allow feedthroughs.

-collinear_margin *distance*

Specifies a distance within which PG routes parallel to (but not overlapping) the block boundary are considered collinear PG routes. See a complete description of this option under **pg_routing** in the **-object_type** description.

-pin_meet_fatwire_rule true | false

Ensures that the size of the PG pin is big enough to trigger fat-wire spacing rules within the block, when this option is true. By default, this option is false. For more information, see the description under **pg_routing**.

DESCRIPTION

This command sets options for the **push_down_objects** command. You can review the options with the **report_push_down_object_options** command.

EXAMPLES

The following example sets *block_action* to *copy* and *create_pin_shape* for *pg_routing* pushing down.

```
prompt> set_push_down_object_options -object_type pg_routing \
    -block_action {copy create_pin_shape}
```

The following example sets *block_action* to *copy* and *create_pin_shape* for *signal_routing* pushing down.

```
prompt> set_push_down_object_options -object_type signal_routing \
```

-block_action {copy create_pin_shape}

The following example sets the block actions for cell pushdown to include retain_obsolete_ports.

prompt> **set_push_down_object_options -object_type cells** \
-block_action {copy retain_obsolete_ports}

The following example sets the "cells" object-type-specific option **-allow_multi_rail_cells**.

prompt> **set_push_down_object_options -object_type cells** \
-allow_multi_rail_cells true

SEE ALSO

push_down_objects(2)
remove_push_down_object_options(2)
report_push_down_object_options(2)

set_pvt_configuration

Sets the allowable PVTs for the current session.

SYNTAX

string **set_pvt_configuration**
[-add]
[-name *new_rule_name*]
[-rule *rule_name*]
[-clear_filter *filter_type*]
[-process_labels *label_list*]
[-process_numbers *number_list*]
[-voltages *voltage_list*]
[-temperatures *temp_list*]

Data Types

filter_type string
label_list list of string
number_list list of float
voltage_list list of float
temp_list list of float
rule_name string
new_rule_name string

ARGUMENTS

-add

Add a new rule to the configuration, making it the current rule, and apply the rest of the arguments to the new rule. This option is exclusive with *-rule*.

-name *new_rule_name*

Used along with -add, optionally gives the new rule a specific name. Without *-name*, new rules receive names like rule_1. This option is exclusive with *-rule*.

-rule *rule_name*

Make *rule_name* the current rule, and apply the rest of the arguments to it. This option is exclusive with *-add*.

-clear_filter *filter_type*

Specifies which filters you want to clear, that is, which of process label, process number, voltage, or temperature you do not want filtered. This is applied to the current rule before any of the other filters. Valid values are:

- **all** Clears all filters
- **process_label** Clears only the process label filter
- **process_number** Clears only the process number filter
- **voltage** Clears only the voltage filter
- **temperature** Clears only the temperature filter

-process_labels ***label_list***

Specifies process labels that you want included in the current rule in the configuration.

-process_numbers ***number_list***

Specifies process numbers that you want included in the current rule in the configuration.

-voltages ***voltage_list***

Specifies voltages that you want included in the current rule in the configuration.

-temperatures ***temp_list***

Specifies temperatures that you want included in the current rule in the configuration.

DESCRIPTION

The **set_pvt_configuration** command is used to limit the process, voltage, and temperature combinations which are allowed in the current session. The concept is the same across all applications using the command, but there are details which vary. These will be described below.

The configuration is a sequence of rules, each of which specifies process labels, process numbers, voltages, and temperatures to match. Generally, if you want all data matching a specific set of PVTs, a single rule is sufficient. When you have a specific subset of many PVTs that you want to match, multiple rules are needed. See examples below. New rules are created with *-add*. If desired, you can name a new rule with *-name*. Otherwise, they are automatically named. You can change the rule that the command is operating on with the *-rule* option. The result of the command is the name of the current rule. If neither *-add* or *-rule* is used, the current rule is used. If no rules exist, an initial rule is created for you. At least one of the filtering options - *-process_labels*, *-process_numbers*, *-voltages*, *-temperatures* and *-clear_filter* - is required if you use *-add* or *-rule*.

By default, all PVTs are allowed. Each of the filters allows a list of values, and these add to previous values already in the configuration. If you want to overwrite a particular filter, you would use the *-clear_filter* option in combinations with the other arguments which change a rule in the configuration. You can also do the same thing in two commands: one to clear, and one to add. Note that rules with no constraints are deleted.

The output from the **set_pvt_configuration** command shows the current configuration. For categories that are unfiltered in a particular rules (that is, for which you have not provided any specific configuration), the output will say that all are included. See the examples below. If you just want to show the current configuration, use the command without any arguments.

Configuration in the Library Manager

In the library manager, the **set_pvt_configuration** command limits the process, voltage, and temperature combinations which a DB or .lib file can have in the session. As these files are read, they are compared to the rules in the configuration to determine if they meet the criteria. If they do not pass, they are excluded from the session. This allows you to declare the needed processes, voltages and temperatures, then cast a wide net over a set of DB or .lib files, without having to depend on a file naming convention, as the rules are applied to the data in the file (or data added to the file when it was read). The voltage values

considered are the nominal values, that is, the values from a default operating condition. Other voltage rails are not considered.

Note that the configuration does not apply to the aggregate, edit, or verification flows.

For all other flows, the **set_pvt_configuration** command can be used before or after **create_workspace**. Changing the configuration after files have been read will enable, disable, or reread files as needed. Changing the process of a library using **set_process** may also have an impact on the configuration.

It is very possible to have a different number of files known to the configuration, available from **get_libs**, or attached to the workspace (visible with **report_workspace**). The **get_libs** command provides all libraries which have been successfully read (and not removed). **report_workspace** shows files which currently match the configuration. Finally, the configuration internally keeps track of "greyed out" files - those which were passed to **read_db** or **read_lib**, may or may not have been read into memory, but do not match the current configuration. There is an option to see the greyed out files in the GUI.

Configuration in an Implementation Tool

In an implementation tool, the **set_pvt_configuration** command limits the process, voltage, and temperature combinations which an NDM reference library can have have in the session. In contrast to the library manager, the configuration must be established before **open_lib** is used. Changes made to the configuration are NOT applied to libraries that are already open. If your current configuration blocks all panes in a particular library, a warning will be issued for that library. If you have created a configuration, then any panes that do match will be listed after **open_lib**. If you have not created a configuration, all panes match, and no messages are listed. If you have created a configuration which is so restrictive that no panes match, a LIB-072 warning is issued. You can always use **report_lib** to show more information about the panes.

EXAMPLES

The following example sets the configuration to allow any process label, any process number, voltages 0.8 and 0.9, and temperatures 0, 125, and -40.

```
prompt> set_pvt_configuration -clear_filter all -voltages {0.8 0.9} -temperatures {0 125 -40}
Current PVT Configuration
 rule_1:
  Process Labels :  <All included>
  Process Numbers:  <All included>
  Voltages       :  0.8, 0.9
  Temperatures   :  -40, 0, 125

rule_1
prompt>
```

To add the limitation for only process labels SS and FF:

```
prompt> set_pvt_configuration -process_labels {SS FF}
Current PVT Configuration
 rule_1:
  Process Labels :  "FF", "SS"
  Process Numbers:  <All included>
  Voltages       :  0.8, 0.9
  Temperatures   :  -40, 0, 125

rule_1
prompt>
```

Now if you read DB files and some are screened out you might see messages like the following, which indicate that the file was not loaded. This would be expected based on the setting from the previous example. The voltage does not match.

```
prompt> read_db [glob ${DBDIR}/*.db]
Information: Loading db file '/path/stdcell_ss_0.815V_0C.db' (FILE-007)
Warning: Did not load /path/stdcell_ss_0.815V_0C.db because its
PVT (label='SS', P=1.0, V=0.815, T=0) does not match the current
configuration (LM-133)
```

If you wanted to allow files which never had a process label applied, you do that with the empty string. Assuming that all files had a process label, you would not see any messages about file activation.

```
prompt> set_pvt_configuration -process_label [list ""]
Current PVT Configuration
 rule_1:
  Process Labels :  "", "FF", "SS"
  Process Numbers:  <All included>
  Voltages       :  0.8, 0.9
  Temperatures   :  -40, 0, 125

rule_1
prompt>
```

Numerical comparisons are made with a very tight tolerance. So 0.8 will not match 0.815.. In the library manager, if you realize that your voltages are really 0.815 and 0.92, you can clear voltages without impact on the rest of the configuration, and apply new ones. Now, the file that was screened out above is already known, so it is re-activated, and in this case, re-read since it was never loaded into memory.

```
prompt> set_pvt_configuration -clear_filter voltage \
   -voltages [list 0.815 0.92]
Current PVT Configuration
 rule_1:
  Process Labels :  "", "FF", "SS"
  Process Numbers:  <All included>
  Voltages       :  0.815, 0.92
  Temperatures   :  -40, 0, 125
Information: Configuration change re-activates file
/path/stdcell_ss_0.815V_0C.db (LM-132)
Information: Loading db file '/path/stdcell_ss_0.815V_0C.db' (FILE-007)

rule_1
prompt>
```

In the library manager, if you were to remove a temperature that matches files, you would see de-activation messages:

```
prompt> set_pvt_configuration -clear_filter temperature \
   -temperatures {125 -40}
 rule_1:
  Process Labels :  "", "FF", "SS"
  Process Numbers:  <All included>
  Voltages       :  0.815, 0.92
  Temperatures   :  -40, 125
Information: Configuration change de-activates file
/path/stdcell_ss_0.815V_0C.db (LM-132)

rule_1
prompt>
```

If you want to capture 0.88 volts but only for "SS", you could add a second rule:

```
prompt> set_pvt_configuration -add -process_label SS -voltages 0.88
 rule_1:
  Process Labels :  "", "FF", "SS"
  Process Numbers:  <All included>
  Voltages       :  0.815, 0.92
  Temperatures   :  -40, 125
 rule_2:
  Process Labels :  "SS"
  Process Numbers:  <All included>
  Voltages       :  0.88
  Temperatures   :  <All included>

rule_2
prompt>
```

Here is an example where multiple rules come into play. Let's say you have these panes in a reference library and you want to use configuration to load panes 1 and 4:

```
 Pane 0: process = "SS" (1),  voltages:  0.6; temperatures: 0
 Pane 1: process = "SS" (1),  voltages:  1.2; temperatures: 0
 Pane 2: process = "SS" (1),  voltages:  1.2; temperatures: 125
 Pane 3: process = "TT" (1),  voltages:  0.6; temperatures: 0
 Pane 4: process = "FF" (1),  voltages:  0.6; temperatures: 0
 Pane 5: process = "FF" (1),  voltages:  0.7; temperatures: 125
 Pane 6: process = "FF" (1),  voltages:  0.9; temperatures: 125
```

You can use these commands and get the configuration shown:

```
 set_pvt_configuration -voltages 1.2 -temperatures 0
 set_pvt_configuration -add -name FF -process_label FF -voltages 0.6

Current PVT Configuration:
 rule_1:
  Process Labels :  <All included>
  Process Numbers:  <All included>
  Voltages       :  1.2
  Temperatures   :  0
 FF:
  Process Labels :  "FF"
  Process Numbers:  <All included>
  Voltages       :  0.6
  Temperatures   :  <All included>
```

If you open this library, you will get the two panes you want:

```
prompt> open_lib p7.ndm
Information: Loading library file 'p7.ndm' (FILE-007)

Panes activated based on PVT configuration (7 possible panes):
  Pane 1: process = "SS" (1),  voltages:  1.2; temperatures: 0
  Pane 4: process = "FF" (1),  voltages:  0.6; temperatures: 0
```

You would not be able to achieve this with a single rule. If you tried to include the above processes, voltage, and temperature in a single rule, it would match panes 0, 1 and 4. This is where multiple rules can help.

SEE ALSO

create_workspace(2)
report_workspace(2)
set_process(2)
read_db(2)
read_lib(2)
remove_lib(2)
open_lib(2)
LM-133(n)
LIB-070(n)
LIB-071(n)
LIB-072(n)
LIB-073(n)

set_qor_strategy

Applies or reports optimization metrics and application settings for the current design.

SYNTAX

status **set_qor_strategy**
[-output *file_name*]
[-report_only]
[-diff_only]
[-reduced_effort]
[-metric {area congestion leakage_power timing total_power} | {timing}
| {leakage_power} | {total_power}]
-stage pnr | synthesis

Data Types

file_name string

ARGUMENTS

-output *file_name*

Specifies the name of the output file name for writing the application options.

-report_only

Writes out the application options to the console in tabular format.

-diff_only

Writes out the application options to the console in tabular format for application options that are not set to their target settings.

-reduced_effort

When specified with one of the single metric targets of timing, leakage_power, or total_power, the -reduced_effort enables a lower effort recipe. This sets the engines to work toward the specified metric target but tradeoff some performance for runtime. This is aimed for designs that are easier to close and where faster runtime is more desirable.

-metric *{area congestion leakage_power timing total_power} | {timing} | {leakage_power} | {total_power}*

Specifies one or more metrics to use when configuring application options. Valid metric names are "area", "congestion", "leakage_power", "timing", and "total_power". When more than one metric is specified, the tool defaults to the old behavior. The metric targets are additive and can be set again. When only one metric is specified, and it is either "timing" or "leakage_power" or "total_power", the tool enacts new behavior. Application options are set to optimize the specified target only.

-stage pnr | **synthesis**

Specifies the stage name. Valid stage names are "synthesis" or "pnr". You can specify only one stage name.

DESCRIPTION

This command applies metric-specific settings to the current design. It sets the application options for the synthesis, placement, and routing engines for better performance in terms of one or more of the following metrics: area, congestion, leakage_power, total_power and timing. By default, the command applies the application option settings to the design based on the **-metric** and **-stage** options, and sets the metric_target attribute on the block.

Use the **-output file_name** or **-report_only** options to preview the application option settings before applying them to the design. The **-output file_name** option writes the application option settings as Tcl commands which can be applied by using the **source** command. The **-report_only** option writes out the application option settings to the console.

The **-reduce_effort** option is for trading off performance for runtime by lowering the efforts of the tool engines. The **set_qor_strategy** command still optimizes toward the single metric target specified in **-metric**.

EXAMPLES

The following example writes the application option settings to the console, but does not change the application option settings, for the "area" metric and the "pnr" stage.

```
prompt> set_qor_strategy -metric area -stage pnr -report_only
Metric(s) : area
+-----------------------------------+---------------+------------+------------+----------+
|         Option Name               | Metric Group  | Tool       | Current    | Target   |
|                                   |               | Default    | Setting    | Setting  |
+-----------------------------------+---------------+------------+------------+----------+
| route.global.timing_driven        | *All metrics* | false      | false      | true     |
| time.enable_clock_to_data_analysis | *All metrics* | false      | false      | true     |
...
```

The following example applies the application option settings for the area, total_power and timing target metrics for the pnr (placement and routing) stage (old behavior).

```
prompt> set_qor_strategy -stage pnr -metric {area total_power timing}
prompt> place_opt
```

The following example applies the application option settings for the total_power target metric for the pnr (placement and routing) stage (new behavior).

```
prompt> set_qor_strategy -stage pnr -metric {total_power}
prompt> place_opt
```

SEE ALSO

set_hpc_options(2)

set_query_rules

Defines query rules.

SYNTAX

status **set_query_rules**
[-hierarchical_separators *separators*]
[-bus_name_notations *bus_notations*]
[-class *class_list*]
[-regsub *regsub*]
[-wildcard]
[-suffix *suffix_list*]
[-nocase]
[-verbose]
[-show]
[-reset]

Data Types

separators list
bus_notations list
class_list list
regsub list
suffix_list list

ARGUMENTS

-hierarchical_separators *separators*

Specifies a list of equivalent hierarchy separators in object names. There must be at least two elements in the list. The default list is {/ _ .}. The elements are preferably all single characters, although at most one multi-character element is allowed. No more than one alphabetical character a-z or A-Z is allowed as a hierarchical separator. The alphabetical character is also mutually exclusive with the multi-character element. The following special characters are not supported as hierarchy separators nor are they allowed as one of the characters for the multi-character hierarchy separator: 0-9, '*', '?', '\\', '+', '^', '[', ']'.

-bus_name_notations *bus_notations*

Specifies a list of equivalent bus name notations in object names. There must be at least two elements in the list, and the list element must be two characters. The first character of the element is the opening bus name character, and the second character is the closing bus name character. The default list is {[] __ ()}. No more than one matching pair of alphabetical characters a-z or A-Z is allowed as bus name notations. The following characters are not supported as opening or closing bus name characters: 0-9, '*', '?', '\\', '+', '^', and '/'. Bracketed bus notations must be paired. For example, "[)" are not properly paired brackets, so they are not supported. For nonbracket bus name notations, the opening and closing bus name character must be the same. For example, "-_" is not a supported bus name notation.

-class *class_list*

Specifies a list of object class names for which the query rules are applied. The following object classes are supported: *cell*, *port*, *pin*, *net*, *power_domain*, *supply_net*, *supply_set*, *supply_port*, and *power_switch*. You can specify a subset of the above object classes. By default, the command applies the query rules to all supported classes.

-regsub *regsub*

Specifies a list of options for the **regsub** Tcl command. Each list should include three string elements {{switches}, {match_regexp}, {substitution_exp}}. The options can be specified multiple times in one **set_query_rules** command. In addition to the Tcl built-in switches, the -class switch {**-class cell** | **pin** | **net** | **port**} is supported to enable object class-specific regsub rules.

This option is applicable to pin, port, net or leaf level cells. In case the option is given for cell, it could also be applied to the cell of a pin. If **-regsub** is used without other query rules, the default values for the other options are still used. In order to save runtime, use this option only if the existing rule-based matching scheme fails to match an object.

-wildcard

Enables wildcard support.

-suffix *suffix_list*

Predefined suffix rules {_reg _tri} could be enabled with this option to match register or tri-state cells. This rule is not applicable to other type of cells.

-nocase

Enables case-insensitive rule-based matching.

-verbose

Enables verbose messages.

-show

Shows the current definition of the query options. When **-show** is used alone,the existing query rules are not cleared in the current design. If used with other options, the new query options will be set first and then displayed.

-reset

Resets query rules to the program default.

DESCRIPTION

The **set_query_rules** command allows you to define query rules. The purpose of the rule-based matching is to resolve naming differences, such as differences in hierarchical separators and bus naming notations.

This command clears all query rules in the current design, and then sets the new query rules in the current design. The new rule settings will replace the previous settings.

EXAMPLES

Show program default.

```
prompt> set_query_rules -show

*************************
Query Rules
*************************

         Rule Type   Rule Value
 -------------------   --------------------
 Hierarchy Separator   {/ _ .}
       Bus Notation   {[] __ ()}
       Object Class   {cell port pin net power_domain power_switch
                      supply_net supply_set supply_port}
            Nocase   false
          Wildcard   false
           Verbose   false
1

# Non-default
prompt> set_query_rules \
   -hierarchical_separators {/ _} \
   -bus_name_notations {[] ||} \
   -class {cell port} -show

*************************
Query Rules
*************************

         Rule Type   Rule Value
 -------------------   --------------------
 Hierarchy Separator   {/ _}
       Bus Notation   {[] ||}
       Object Class   {cell port}
            Nocase   false
          Wildcard   false
           Verbose   false
1

# Use alphabetical characters
prompt> set_query_rules \
   -hierarchical_separators {/ x} \
   -class {cell port} \
   -bus_name_notations {[] __ || xx} -show

*************************
Query Rules
*************************

         Rule Type   Rule Value
 -------------------   --------------------
 Hierarchy Separator   {/ x}
       Bus Notation   {[] __ || xx}
       Object Class   {cell port}
            Nocase   false
          Wildcard   false
           Verbose   false
1
```

```
# Setting regsub rules
prompt> set_query_rules -show \
   -class {port cell pin net}  \
   -regsub {{-class cell} {(.*)\[(\d+)\]} {\1_\2} } \
   -regsub {{-class net} {RED} {3}}

*************************
Query Rules
*************************

          Rule Type   Rule Value
 -------------------   ----------------------
 Hierarchy Separator   {/ _ .}
       Bus Notation   {[] __ ()}
       Object Class   {cell port pin net}
            regsub   {{-class cell} {(.*)\[(\d+)\]} {\1_\2}}
            regsub   {{-class net} {RED} {3}}
             Nocase   false
          Wildcard   false
           Verbose   false
1
```

SEE ALSO

define_name_maps(2)
write_name_map(2)
read_name_map(2)
extract_physical_constraints(2)
design.enable_rule_based_query(3)
get_cells(2)
get_pins(2)
get_nets(2)
.prod

set_rail_scenario

Associates a rail result with a design scenario.

SYNTAX

status **set_rail_scenario**
-design_scenario *design_scenario*
-result_name *rail_result_name*
[-percentage_threshold *percentage_threshold*]
[-priority_weight *priority_weight*]

Data Types

string *design_scenario*
string *rail_result_name*
float *percentage_threshold*
float *priority_weight*

ARGUMENTS

-design_scenario *design_scenario*

Specifies a design scenario name to be associated with a rail result.

-result_name *rail_result_name*

Specifies a rail result name to be associated with a design scenario.

-percentage_threshold *percentage_threshold*

Specifies a percentage threshold such that any voltage drop ratio (voltage drop divided by ideal supply voltage) greater than this threshold is considered a violation.

-priority_weight *priority_weight*

Specifies a priority weight to be coupled to the voltage drop ratio. Set a higher priority weight to the smaller voltage drop values such that these voltage values are shown with similar priority to those with bigger voltage drop values. Without the priority weight setting, small voltage drop values from one rail result might be totally ignored during optimization.

DESCRIPTION

When multiple rail results are loaded to memory, the command associates a design scenario with a rail result name. By default, all

results are associated with the current scenario if users does not explicitly associate a result with a design scenario with the **set_rail_scenario** command.

Through the command, users can also specify the voltage drop threshold and priority weight values for rail results. Voltage drop threshold and the priority weight are used during optimization. Voltage drop threshold determines which cells are of violations, while priority weight determines priority when cells are optimized when specified together with the voltage drop threshold.

This command has to be invoked after multiple rail results are loaded to memory through the **open_rail_result** command.

EXAMPLES

The following example loads three different rail results, then sets the association between each of the rail results and a design scenario. Since static result is loaded with two dynamic results, a higher priority weight is set on static result, so it will be treated equally with dynamic results during optimization.

```
prompt> open_rail_result { static_result vcd_result vector_free_result } \P
prompt> set_rail_scenario -design_scenario fast -result_name
      static_result -percentage_threshold 0.02 -priority_weight 5.0
prompt> set_rail_scenario -design_scenario typical -result_name
      vcd_result -percentage_threshold 0.1 -priority_weight 1.0
prompt> set_rail_scenario -design_scenario slow -result_name
      vector_free_result -percentage 0.1 -priority_weight 1.0
```

SEE ALSO

open_rail_result(2)
report_rail_scenario(2)

set_ref_libs

Sets the reference library list for a design library.

SYNTAX

set_ref_libs
[-library *name*]
-ref_libs {*paths*}

set_ref_libs
[-library *name*]
-ref_libs {*paths*}
-use_technology_lib *path*

set_ref_libs
[-library *name*]
-add *path*
[-before *path*]

set_ref_libs
[-library *name*]
-remove *path*

set_ref_libs
[-library *name*]
-clear

set_ref_libs
[-library *name*]
-rebind

ARGUMENTS

-library *name*

Which library to set the reference library list. The library must be open. If no library is specified, the current library is used.

-ref_libs {*paths*}

A list of reference library paths or names to set on this library. The entry can be either library paths or library names. A library can have an in-memory name that is different from its path. If a library name different from its path is specified, it is resolved with libraries already in-memory, and the full path of the library, not the in-memory name, will be set on the library.

Besides full pre-build reference libraries, this option also accept physical source data, like frame libraries, LEF files and Milkyway libraries. The tool will automatically build reference libraries based on these physical source data.

-use_technology_lib ***path***

Specify one library on the ref_lib list as a dedicated technology library. To be used in combination with the -ref_libs option. It is an error if the specified technology library does not have a technology section.

-add ***path***

Adds *path* to the end of the library's reference library list.

-before ***path***

Places the **-add path** before the named *path* in the reference library list.

-remove ***path***

Remove the named *path* from the reference library list. This also marks all open blocks of the library as needed to be bound.

-clear

Remove all paths from the reference library list. This also marks all open blocks of the library as needed to be bound.

-rebind

Rebind each reference library path to libraries currently loaded in memory using the search_path.

DESCRIPTION

The **set_ref_libs** command modifies the list of libraries searched for design objects referenced from this library. This is a list of paths used in conjunction with the library's parent directory, and the **search_path** to find libraries. Paths may be library names, partial paths to libraries, or full path names. If it is a partial path, it is resolved using the current search_path setting, except when the partial path is relative, path that starts with "./" or "../", in which case it is resolved relative to the parent directory of the library.

The library's reference library path list can be set with the **-ref_libs** option directly. Alternatively, it can also be automatically created and set by specifying physical source data with the **-ref_libs** option. The supported physical source data includes frame libraries, LEF files and Milkyway libraries. When physical source data is specified with **-ref_libs** option, it will create reference libraries according to these physical source data and the link_library setting. The created reference libraries will be put in the directory indicated by the **lib.configuration.local_output_dir** application option. The **lib.configuration.default_flow_setup** application option can be used to point to a Tcl script with customized settings for the reference library creation. For more information about how to use these application options, see their man pages.

A ref lib list may not contain multiple libraries that resolve to the same name in the tool. For example, two different source libraries with the same name, say <myPath1>/myLib and <myPath2>/myLib where myPath1 and myPath2 are different are not allowed in the same ref lib list. The command will fail if the result of the operation would add duplicate lib names. Furthermore, by convention the tool does not treat most .nlib/.ndm suffixes as part of the lib name, so libraries that differ only in a .nlib/.ndm suffix are subject to the duplicate name restriction. In the previous example, if the given libraries are <myPath1>/myLib and <myPath2>/myLib.ndm they are still not allowed in the same ref lib list as the name myLib.ndm resolves to myLib.

When the reference library list is set, each path is bound to a currently open library on the search path with that name. This binding is recreated when the library is re-opened in a later session. If the **search_path** is changed, the bindings can be updated with **set_ref_libs -rebind**.

Note that rebinding the reference library list with **-rebind** does not affect the bindings of any blocks in memory. Block references within a design can be rebound with **link_block -rebind**.

The list of ref_libs on a library can be retrieved with the get_attribute command.

EXAMPLES

The following example sets two reference libraries on the current library, adds a third path, then removes one path.

```
prompt> set_ref_libs -ref_libs {r4000 macros/TSMC24}
r4000 macros/TSMC24
prompt> set_ref_libs -add r6000 -before TSMC24
r4000 r6000 macros/TSMC24
prompt> set_ref_libs -remove r4000
r6000 macros/TSMC24
prompt> get_attribute [get_libs mylib] ref_libs
r6000 macros/TSMC24
prompt>
```

SEE ALSO

create_lib(2)
current_lib(2)
get_attribute(2)
link_block(2)
open_lib(2)
report_ref_libs(2)
search_path(3)
link_library(3)
lib.configuration.local_output_dir(3)
lib.configuration.default_flow_setup(3)

set_reference

Changes the reference for the specified cells.

SYNTAX

integer **set_reference**
[-design *design*]
[-block *new_block*]
[-to_block *new_block*]
[-to_module *new_module*]
[*cells*]
[-of_object *old_block*]
[-reference *reference*]
[-pin_map *name_map*]
[-pin_rebind safe | force | test | none]
[-pin_verbose]

Data Types

design collection
new_block collection
new_module string
cells collection
old_block collection
reference string
name_map list

ARGUMENTS

-design *design*

Specifies the top-level design in which to look for the specified objects.

If you do not specify this option, the command looks for the objects in the current design.

-block *new_block*

Specifies the new block or library cell to which to bind the specified cells. The argument can be a name or a collection of one object. If you specify the block name, use the following format:

[*lib*:]*block*[/*label*][.*view*]

The **-block** has been deprecated. Please use option **-to_block** instead.

-to_block *new_block*

Specifies the new block or library cell to which to bind the specified cells. The argument can be a name or a collection of one object. If you specify the block name, use the following format:

[*lib*:]*block*[/*label*][.*view*]

The **-to_block** and **-to_module** options are mutually exclusive; you must specify one.

-to_module ***new_module***

Specifies the new module to which to bind the specified cells. The argument must be the name of one object.

The **-to_block** and **-to_module** options are mutually exclusive; you must specify one.

cells

Binds the new reference to the specified cells. The argument can be a list of cell names or a collection of cells.

The *cells* argument, **-of_object** option, and **-reference** option are mutually exclusive; you must specify one.

-of_object ***old_block***

Binds the new reference to all cells in the current block that reference the specified block or library cell. The argument can be a name or a collection of one object.

The *cells* argument, **-of_object** option, and **-reference** option are mutually exclusive; you must specify one.

-reference ***reference***

Binds the new reference to all cells bound to the specified reference. Use the following format to specify the reference name:

[*lib*:]*block*[/*label*]

The reference name cannot include the view name.

The *cells* argument, **-of_object** option, and **-reference** option are mutually exclusive; you must specify one.

-pin_map ***name_map***

Specifies the pin name mappings between the old block and the new block. Use the following format to specify the pin mapping:

{*old_pin1 new_pin1 old_pin2 new_pin2* ...}

The list must have an even number of entries. Each entry must be unique (you cannot map multiple old pins to the same new pin). To remove a pin, specify a new pin name of "".

You can also use a Tcl associative array to specify the name mapping. For an example, see the EXAMPLES section.

-pin_rebind safe | **force** | **test** | **none**

Specifies how to handle pin rebinding.

Valid values are

- **safe** (the default)
  Changes the reference for a cell only when the change does not drop any pins that have a net connection.
- **none**
  Changes the reference for a cell only if the old and new port interfaces are identical.
- **force**
  Always changes the cell reference and drops net connections that do not map to the new reference.
- **test** Checks the pin interface, but does not change the reference for the cell.

-pin_verbose

Displays the old-to-new pin mapping for the cell before the cell reference is changed.

Use this option with the **-pin_rebind test** option to describe the changes that would be made to the pins if the **set_reference** command changed the cell's reference.

DESCRIPTION

The **set_reference** command changes the reference, and optionally the view, of the specified cells. Use one of the following methods to specify the cells:

- Specify the cells explicitly by using the *cells* argument.
- Specify the block or library cell referenced by the cells by using the **-of_object** or **-reference** option.

All the specified cells must have the same current reference, which differs from the new reference.

The command binds the specified cells to the specified reference or module. If you specify the view for the new reference, the command changes the cell view. The command does not use the view switch list, which is specified by the **set_view_switch_list** command, to select the view for the cells.

The command maps cell pins to reference ports by name. To specify the pin mapping, use the **-pin_map** option. By default, the command updates the cell pins to match the reference ports if no cell pin net connections are lost. If a cell pin net connection would be lost, the command issues a warning does not change that cell's reference. To change the pin rebinding method, use the **-pin_rebind** option.

Once all the invocations of **set_reference** are done, a follow up call to the command **"link_block -force" is to be done in some cases. This should be done when the block to which the cell is to be changed is either of module or blackbox design type. The following are the cases where a follow up call to command "link_block -force" is not required.**

- changing the reference from a lib-cell to another lib-cell.
- changing the reference from a design to module.
- changing the reference from a module to design.

EXAMPLES

The following example changes the reference of the u1/u7 cell to alu_1.

```
prompt> set_reference -to_block [get_blocks alu_1] u1/u7
1
```

The following example changes the reference of the specified cells to the abstract view of the xLib:RAMctrl block.

```
prompt> set_reference -to_block [get_blocks xLib:RAMctrl.abstract] \
  {uLL/uRAMctrl1 uLR/uRAMctrl14 uTOP/uRAMctrl7}
3
```

The following example changes the reference of all cells with a reference of cellLib/NAND2X2 to cellLib/NAND2X4.

```
prompt> set_reference -to_block cellLib/NAND2X4 \
```

```
  -of_object [get_lib_cells cellLib/NAND2X2]
14573
```

The following example changes the reference of all cells with a reference of cellLib/INVx16 to the timing view of cellLib:INVx2.

```
prompt> set_reference -to_block cellLib:INVx2.timing \
  -of_object [get_lib_cells cellLib/INVx16]
101277
```

The following example changes the reference of all cells that reference the MID block in the current library to the MID block in the newLib library.

```
prompt> set_reference -to_block newLib:MID -reference MID
2
```

The following example changes the reference of all cells that reference the MID block in the myLib library to the MID block in the newLib library.

```
prompt> set_reference -to_block newLib:MID -reference myLib:MID
2
```

The following example uses pin mapping specified with a Tcl associative to update the pin connections while changing the reference of the cell. Note that the pins change both name and order.

```
prompt> set pinMap(aa) AA
AA
prompt> set pinMap(bb) BB
BB
prompt> set pinMap(cc) CC
CC
prompt> set_reference -to_block newLib:MID.design u2/uMID \
  -pin_map [array get pinMap] -pin_verbose
Pin mapping from 'myLib:MID' to 'newLib:MID.design':
 Pin   0 i1 stays at pin   0.
 Pin   1 i2 stays at pin   1.
 Pin   2 i3 stays at pin   2.
 Pin   3 i4 stays at pin   3.
 Pin   4 i5 stays at pin   4.
 Pin   5 i6 stays at pin   5.
 Pin   6 i7 stays at pin   6.
 Pin   7 o1 stays at pin   7.
 Pin   8 aa moved to pin   9 renamed to 'AA'.
 Pin   9 bb moved to pin   8 renamed to 'BB'.
 Pin  10 cc stays at pin  10 renamed to 'CC'.
1
```

The following example shows the results for the various pin rebinding modes when the port interface differs between the old and new reference.

```
prompt> set_reference -to_block newLib:MID u2/uMID \
  -pin_rebind none
Error: Block 'newLib:MID.design' port interface is different from 'myLib:MID'.
(NDMUI-088)
Error: problem in set_reference
       Use error_info for more info. (CMD-013)

prompt> set_reference -to_block newLib:MID u2/uMID \
  -pin_rebind safe
Warning: Skipping cell 'u2/uMID' to avoid dropping cell pin net connection.
```

```
(NDMUI-089)
0

prompt> set_reference -to_block newLib:MID u2/uMID  \
  -pin_rebind test -pin_verbose
Pin mapping from 'myLib:MID' to 'newLib:MID.design':
  Pin    0 i1 stays at pin    0.
  Pin    1 i2 stays at pin    1.
  Pin    2 i3 stays at pin    2.
  Pin    3 i4 stays at pin    3.
  Pin    4 i5 stays at pin    4.
  Pin    5 i6 stays at pin    5.
  Pin    6 i7 stays at pin    6.
  Pin    7 o1 stays at pin    7.
  Pin    8 aa removed.
  Pin    9 bb removed.
  Pin   10 cc removed.

Removing pin 'u2/uMID/aa' would drop connection to net 'u2/nx1'.
Removing pin 'u2/uMID/bb' would drop connection to net 'u2/nx2'.
0

prompt> set_reference -to_block newLib:MID u2/uMID -pin_rebind force
Information: Disconnecting net 'u2/nx1' from pin 'u2/uMID/aa'. (NDMUI-097)
Information: Disconnecting net 'u2/nx2' from pin 'u2/uMID/bb'. (NDMUI-097)
1
```

SEE ALSO

change_abstract(2)
change_reference(2)
change_view(2)
commit_block(2)
get_blocks(2)
get_cells(2)
get_lib_cells(2)
get_modules(2)
get_view_switch_list(2)
link_block(2)
set_view_switch_list(2)
size_cell(2)
uncommit_block(2)
uniquify(2)
cell_attributes(3)
array(n)

set_register_merging

Sets the **register_merging** attribute on the specified cells or modules, allowing register merging optimization on the objects.

SYNTAX

status **set_register_merging**
obj_list
[true | false]

Data Types

obj_list list

ARGUMENTS

obj_list

Specifies a list of cell or module names for which to enable register merging optimization. Cell names in *obj_list* must be from the current design. If multiple objects are specified, they must be enclosed in quotation marks ("") or braces ({}).

true | **false**

Specifies the value with which to set the **register_merging** attribute. When this option is set to true (the default), register merging is enabled on the specified objects (cells or modules). Register merging is performed on the specified objects provided that the application option **compile.seqmap.enable_register_merging** is set to true.

A setting of false on a module disables register merging for the whole module. A setting of false on an instance disables register merging for the whole instance.

DESCRIPTION

The **set_register_merging** command sets the **register_merging** attribute on the specified *obj_list*. This attribute is used to specify that the cells or modules are to be optimized using register merging.

If a cell with a specified name is found in the current design, the **register_merging** attribute is set to the specified value on the cell. If no cell with a specified name is found, the tool searches for a module and the attribute set on the module.

EXAMPLES

The following example shows how to enable register merging optimization during optimization for the sequential cells named U0 and U1 and how to disable register merging for the U2 cell:

```
prompt> set_register_merging {U0 U1} true
prompt> set_register_merging U2 false
```

SEE ALSO

compile(2)
get_attribute(2)

set_register_output_inversion

Instructs the tool to force or prevent sequential output inversion on the specified objects.

SYNTAX

status **set_register_output_inversion**
obj_list
[true | false]

Data Types

obj_list list

ARGUMENTS

obj_list

Specifies a list of registers or module names for which to control the register output inversion. Cell names in obj_list must be from the current design. If multiple objects are specified, they must be enclosed in quotation marks ("") or braces ({}).

true*|*false

When the value is true, sequential output inversion is forced. When the value is false, sequential output inversion is prevented.

DESCRIPTION

The set_register_output_inversion command Instructs the tool to force or prevent sequential output inversion on the specified objects.

The command overrides the settings on the global app option compile.seqmap.enable_output_inversion

EXAMPLES

For the example below sequential output inversion will be forced on all u1/count_reg* instances

prompt> **set_register_output_inversion [get_cells {u1/count_reg*}] true**

For the example below sequential output inversion will be prevented on u1/A1_reg

```
prompt> set_register_output_inversion u1/A1_reg false
```

SEE ALSO

set_register_replication

Sets the **register_replication** attribute on the specified sequential cells, thus allowing register replication on the objects.

SYNTAX

int **set_register_replication**
[-max_fanout *max_fanout_value*]
[-num_copies *copy_value*]
[-include_fanin_logic *cell*]
[-include_fanout_logic *cell*]
[-driven_by_original_register *end_points_list*]
object_list

Data Types

max_fanout_value integer
copy_value integer
cell string
end_points_list list
object_list list

ARGUMENTS

-max_fanout *max_fanout_value*

Specifies the maximum fanout of the register after the replication.

-num_copies *copy_value*

Specifies the number of copies the specified register is replicated.

-include_fanin_logic *cell*

Specifies the combinational cell in the fanin of the register for path replication.

-include_fanout_logic *cell*

Specifies the combinational cell in the fanout of the register for path replication.

-driven_by_original_register *end_points_list*

Specifies the endpoints objects (registers or primary output ports) that should remain connected to the original register.

object_list

Specifies the set of registers to replicate.

DESCRIPTION

Sets the **register_replication** attribute on *object_list*. This attribute is used to specify how many copies the sequential cells are replicated. When you specify the **-num_copies** option, the **register_replication** attribute value is the value of the **-num_copies** option. Otherwise, it is the value of the **-max_fanout** option.

The **-include_fanin_logic** and **-include_fanout_logic** options can be used to replicate a path. A single combinational cell is specified for these options and denotes the start or end of the path. Only a single object can be specified in the **object_list** when **-include*** options are used.

The **-driven_by_original_register** option can be used to keep specified endpoints objects (registers or primary output ports) connected to the original register when replicating registers.

If a sequential cell with a specified name is found in the *current design*, the **register_replication** attribute is set to the specified value in the cell.

EXAMPLES

The following example shows how to replicate sequential cell U0 three times and how to replicate sequential cell U1 so that its fanout is less than or equal to four.

```
prompt> set_register_replication -num_copies 3 U0
prompt> set_register_replication -max_fanout 4 U1
```

The following example shows how to get an additional copy of the path from the A_reg register to the U1 cell, which lies in the fanout of the register:

```
prompt> set_register_replication -num_copies 2 A_reg -include_fanout_logic U1
```

SEE ALSO

compile(2)
get_attribute(2)

set_regular_multisource_clock_tree_options

It sets options for command synthesize_regular_multisource_clock_trees to perform auto tap synthesis and global clock tree synthesis for regular multisource clock trees.

SYNTAX

status **set_regular_multisource_clock_tree_options**
-clock *clock*
[-net *net*]
[-tap_template_cells *cells*]
-topology htree_only
-tap_boxes *integer_pair*
[-tap_lib_cells *lib_cell_list*]
-htree_lib_cells *lib_cell_list*
[-tap_boundary *poly_rect*]
[-prefix *string*]
[-max_displacement *distance*]
[-keepouts *poly_rect_list*]
[-htree_layers *layer_list*]
[-htree_routing_rule *routing_rule*]

Data Types

clock a clock in the design
net a clock net in the block
cells list of cell instances
integer_pair list of two positive integers {columns rows}
distance distance in user units
poly_rect a polygon definition (see create_poly_rect for format)
poly_rect_list list of poly_rect
lib_cell_list list of lib_cells
string a string value
layer_list list
routing_rule string

ARGUMENTS

-clock *clock*

This is a required switch. Specifies the clock on which auto tap synthesis must be performed. The synthesis command will use the given clock to get clock related constraints and sinks.

-net *net*

This is optional switch. If -net is not specified then root net of clock specified with -clock is picked up as the net for autotap synthesis. Use this switch if autotap synthesis must be run on a net different than the clock root net. When -tap_template_cells are not present, the tap drivers are inserted on this net. When -tap_template_cells are given, the net should be on the template cell's input pin and H tree will be built on this net.

-topology *htree_only*

This is a required switch. Specifies the type of the global clock distribution structure on top of the tap drivers. Use htree_only when topology includes Htree directly driving tap drivers. Tap synthesis will perform Htree feasibility check. The switch will be extended to support other topologies like clock mesh and clock mesh with Htree in furture

-tap_boxes *integer_pair*

This is a required switch. The **-tap_boxes** option accepts a list of two positive integers defining the number of columns and rows of a location grid. The bounding box of the boundary is subdivided in equally sized boxes according to the specified number of columns and rows. Then the preferred locations are formed by the centers of these boxes. The boundary is derived from the sink distribution rooted from the net given by -net option. The exception is when it is specified explicitly using the **-tap_boundary** option.

-tap_lib_cells *lib_cell_list*

This is a required switch. Specifies a list of buffer lib_cells that can be used as tap driver cells. All specified lib_cells should have lib_cell purpose **cts** included. In case multi voltage support is required single and dual rail versions of the repeaters can be given. For each preferred location, the corresponding voltage area is queried, and the **first** lib_cell in the list that qualifies for that voltage area will be used for the location. If none of the provided driver types qualifies, then no buffer is inserted at that location. This options is mutually exclusive with the **-tap_template_cells** option.

-htree_lib_cells *lib_cell_list*

This is a required switch. Specifies a list of library cells of type buffer or inverter which are allowed to be used during synthesis. They need not have any purpose assigned but should not be marked dont_touch. In case multi voltage support is required single and dual rail versions of the repeaters can be given. The order of the given library cells play a role. The first, second etc. occurrence of a single rail buffer/inverter is matched with first, second etc. occurrence of a dual rail buffer/inverter. It is assumed that these pairs of repeaters are electrical equivalent in terms of input capacitance and delay. Any single or dual rail repeater which is not matched with a counterpart is dropped.

-tap_boundary *poly_rect*

This is optional switch. Specifies the area in which the inserted cells should be placed. By default, boundary is derived from the sink distribution rooted from the net specified. This switch can be used to specify a bodundary different than the default value. If the preferred location of any of the cells to be inserted is not covered by the area enclosed by the boundary, then the cell will not be inserted.

-prefix *string*

This is optional switch. All newly created cells will get this prefix in front of their name. This same prefix is also applied to the names of new nets or ports that are created for a clock driver. The default value is **clk_drv** for tap drivers and msgts for htree drivers.

-max_displacement *distance*

This is optional switch. The inserted clock drivers are legalized taking fixed cells, macros, blockages, and other clock cells into account. When after legalization the distance of the cell center to the preferred location is larger than the distance specified with the **-max_displacement** option, no driver will be inserted near that preferred location. By default, no maximum displacement is in effect, and cells will be legalized as close as possible to the preferred location. This impacts only tap driver insertion

-keepouts *poly_rect_list*

This is optional switch. Specifies a list of poly_rects where no tap driver driver cells should be placed. If a preferred location is in one of the poly_rects, then no driver will be placed at that location.

-htree_layers *layer_list*

This is optional switch. Specifies the layers used in routing the H tree. If not specified then command will pick the layers from clock tree settings

-htree_routing_rule *routing_rule*

This is optional switch. Specifies the non-default routing rule used in routing the H tree. If not specified then command will pick the NDR from clock tree settings

DESCRIPTION

The *set_regular_multisource_clock_tree_options* command takes user options for performing regular multisource clock tree synthesis. To do multiple clock synthesis using synthesize_regular_multisource_clock_trees, the order of the synthesis will follow the order of calls to this option command when -clocks under the synthesis command is not given.

Multicorner-Multimode Support

This applies the settings to clock in current_mode.

EXAMPLES

The following example inserts tap drivers in 4 columns and 2 rows for clock clk ensuring Htree compatibility and builds Htree automatically to drive the tap drivers using specified library cells, NDR and layer list.

```
prompt> set_regular_multisource_clock_tree_options -clock clk -topology htree_only -prefix MSCTS \
 -tap_boxes {4 2} -tap_lib_cells [get_lib_cells */CKBUF*] \
 -htree_lib_cells [get_lib_cells */CKINV*] -htree_layers "m9 m10" -htree_routing_rule "htree_ndr"

prompt> synthesize_regular_multisource_clock_trees
```

SEE ALSO

synthesize_regular_multisouce_clock_trees(2)
report_regular_multisouce_clock_tree_options(2)
remove_regular_multisouce_clock_tree_options(2)
remove_clock_drivers(2)
remove_multisource_global_clock_trees(2)
set_multisource_clock_tap_options
synthesize_multisource_clock_taps
create_routing_rule(2)
set_routing_rule(2)
set_clock_routing_rules(2)
cts.multisource.enable_full_mv_support

cts.multisource.verbose
cts.multisource.enable_pin_accessibility_for_global_clock_trees
place_opt.flow.enable_multisource_clock_trees

set_related_supply_net

Associates supply net(s) to the port of the design or the pin of a cell.

SYNTAX

status **set_related_supply_net**
[-object_list *objects*]
[-ground *ground_supply_net_name*]
[-power *power_supply_net_name*]

Data Types

objects	list
ground_supply_net_name	string
power_supply_net_name	string

ARGUMENTS

-object_list *objects*

Specifies the list of ports (pins) to be associated with power/ground nets or reset. If this option is not specified, all the ports are considered.

-ground *ground_supply_net_name*

Specifies the name of the ground supply net.

-power *power_supply_net_name*

Specifies the name of the power supply net.

DESCRIPTION

This command is used to associate supply nets with design ports or leaf pins. The tool uses this information for constraining and checking the design. The level-shifter insertion tool also uses supply nets to determine if level shifter is required between a port (pin) and the logic connected to the port (pin).

If this command is not specified, the tool assumes that ports are externally connected to the primary supply net of the domain of the top design.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example sets the *VDD_EXT* power supply net and *VSS_EXT* ground supply net with *INPUT* port.

```
prompt> set_related_supply_net -object_list\
[get_ports INPUT] -power VDD_EXT -ground VSS_EXT
```

SEE ALSO

get_ports(2)

set_repeater

Defines the UPF repeater strategy for the power domains in the design.

SYNTAX

status **set_repeater**
repeater_strategy_name
-domain *power_domain*
[-repeater_supply *repeater_supply_set*]
[-applies_to inputs | outputs | both]
[-applies_to_boundary upper | lower | both]
[-elements *objects*]
[-exclude_elements *exclude_objects*]
[-name_prefix *prefix*]
[-name_suffix *suffix*]
[-update]

Data Types

repeater_strategy_name string
power_domain string
repeater_supply_set string
objects list
exclude_objects list
prefix string
suffix string

ARGUMENTS

repeater_strategy_name

Specifies the name of the UPF repeater strategy. The name of the repeater strategy should be unique within the specified power domain.

-domain *power_domain*

Specifies the name of the power domain to which this UPF repeater strategy will be applied.

-repeater_supply *repeater_supply_set*

Specifies the supply set whose power and ground functions are to be used as the repeater power and ground nets respectively.

-applies_to inputs | **outputs** | **both**

Specifies whether the given **set_repeater** command applies to all inputs or outputs or both types of ports of the power domain.

-applies_to_boundary upper | lower | both

Specifies the domain boundary that the given **set_repeater** command applies to. The default value for this option is **both** for a domain with the lower domain-boundary design attribute on, and **upper** for domains that do not have the lower domain-boundary design attribute set or have the lower domain-boundary attribute set to off.

-elements *objects*

Specifies the objects to which this UPF repeater strategy will be applied. The objects can be pins of the root cells of the power domain or top level ports or root cells of the power domain.

-exclude_elements *exclude_objects*

Specifies the objects to which this UPF repeater strategy won't be applied. The objects can be pins of the root cells of the power domain or top level ports or root cells of the power domain.

-name_prefix *prefix*

Specifies the prefix to use for the repeater cell names.

-name_suffix *suffix*

Specifies the suffix to use for the repeater cell names.

-update

Indicates that this command adds *elements* or *exclude_elements* for a previous command with the same *strategy_name* and *domain_name* and executed in the same scope.

With the **-update** option, the command only accepts **-elements** or **-excluded_elements**, and the required option *strategy_name* and **-domain**. If other options are specified with **-update**, the command will error out.

DESCRIPTION

This command defines the UPF repeater strategy for the ports of the specified power domain.

If the **-elements** and **-applies_to** options are not specified, then the repeater strategy will be applied to all ports of the power domain.

If **-repeater_supply** is not specified, then the command will error out.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to define a UPF repeater strategy for all output ports of the specified power domain PD1.

```
prompt> set_repeater repeater_1 -domain PD1 -repeater_supply PD1.primary -applies_to output\
```

The following example shows how to define a UPF repeater strategy for specific ports of the specified power domain:

```
prompt> set_repeater repeater_2 -domain PD1 \
        -repeater_supply rptr_supply_set \
 -elements inst/special_port
```

The following example shows how to define a UPF repeater strategy using **-exclude_elements**:

```
prompt> set_repeater repeater_3 -domain PD1 \
        -repeater_supply rptr_supply_set \
 -elements inst \
        -exclude_elements inst/special_port
```

SEE ALSO

set_port_attributes(2)

set_repeater_group

Define a group of repeaters.

SYNTAX

status **set_repeater_group**
-group_id *group_id*
[-cells *cell_list*]
[-cutline *{ {{x1 y} {x2 y}} | {{x y1} {x y2}} }*]
[-driver_group_id *group_id*]
[-path_drivers *pin_port_list*]
[-path_loads *pin_port_list*]
[-override]

Data Types

group_id	integer
cell_list	collection
pin_port_list	collection

ARGUMENTS

-group_id *group_id*

Specifies the group ID for a group of repeaters.

-cells *cell_list*

Specifies the list of repeaters belonging to the group.

-cutline *{ {{x1 y} {x2 y}} | {{x y1} {x y2}} }*

Specifies the cutline of the group of repeaters.

-driver_group_id *group_id*

Specifies the driver group ID when there is a chain of repeater groups. For example, the first group in the chain is considered the driver group for the second group.

-path_drivers *pin_port_list*

Specifies the macro pins or top ports as drivers and connect to the cells in the first group when there is a chain of repeater groups.

-path_loads *pin_port_list*

Specifies the macro pins or top ports as loads and connect to the cells in the last group when there is a chain of repeater groups.

-override

Used to override all cells in the cell list of an existing repeater group. Must be specified with **-cells**.

DESCRIPTION

This command defines a group of repeaters for cutline support.

EXAMPLES

The following examples show the usages of command.

```
prompt> set_repeater_group -group_id 1 -cells $buf_group_1
-cutline {{250  90} {250 105}} -path_drivers $driver_pins
prompt> set_repeater_group -group_id 2 -cells $buf_group_2
-cutline {{190 130} {205 130}} -driver_group_id 1
prompt> set_repeater_group -group_id 3 -cells $buf_group_3
-cutline {{165 155} {165 175}} -driver_group_id 2
prompt> set_repeater_group -group_id 4 -cells $buf_group_4
-cutline {{110 155} {110 175}} -driver_group_id 3 -path_loads $load_pins_group_4
prompt> set_repeater_group -group_id 5 -cells $buf_group_5
-cutline {{135 220} {150 220}} -driver_group_id 3 -path_loads $load_pins_group_5
```

The following examples shows usages of overriding cells of an existing repeater group.

```
prompt> set_repeater_group -group_id 6 -cells $buf_group_6
-cutline {{145 230} {155 230}} -path_drivers $driver_pins6
prompt> set_repeater_group -group_id 6 -cells $buf_group_7 -override
```

SEE ALSO

report_repeater_groups(2)
place_group_repeaters(2)

set_repeater_group_constraints

Sets constraints for a group of repeaters.

SYNTAX

status **set_repeater_group_constraints**
-type *type_list*
[-horizontal_repeater_spacing] *spacing*
[-vertical_repeater_spacing] *spacing*
[-layer_cutting_distance *layer_scale_list*]

Data Types

type_list list
spacing integer
layer_scale_list list

ARGUMENTS

-type *type_list*

Specifies the types of repeater group constraints, The valid values are, "placement" and "insertion". Use this command to set the constraints, globally. However, constraints set using this command has lower priority than the equivalent command options in the add_group_repeaters and place_group_repeaters commands.

-horizontal_repeater_spacing *spacing*

Specifies the horizontal spacing between two adjacent repeaters in the horizontal direction in terms of number of site width. For example, if the width of the repeater is 9 cell site, then the minimum horizontal spacing is 9 um. When you specify the spacing to be less than 9, this option automatically extends the spacing to 9 um. To achieve the checkerboard pattern, in this example, the horizontal spacing should be 9,13,17,21....

-vertical_repeater_spacing *spacing*

Specifies the vertical spacing between repeaters in the vertical direction in terms of number of site height. When cell is single site height, the minimum vertical spacing is 1 site height. To achieve a checkerboard pattern, in this example, the vertical spacing should be 1,5,9.... When you specify the vertical spacing as 4, this option automatically rounds it to 5. For a double site height library cell, to ensure checkerboard pattern, you can specify 2,6,10....

-layer_cutting_distance *layer_scale_list*

Specifies the layer cutting distance list. Using this option, you can define input routes and output routes to cell center distances. Specify the option by using the following format: {{layer1 cutting_distance_in_1 cutting_distance_out_1} {layer2 cutting_distance_in_2 cutting_distance_out_2} ...}

DESCRIPTION

This command defines the constraints for a group of repeaters. Constraints set using the set_repeater_group_constriants command has lower priority than the corresponding command options in add_group_repeaters and place_group_repeaters commands. Constraints set using the set_repeater_group_constriants command for insertion and placement are independent. You can overwrite the constraints using another command run.

EXAMPLES

The following examples show the usages of command.

prompt> **set_repeater_group_constraints -type insertion**
-horizontal_repeater_spacing 10 -vertical_repeater_spacing 3
-layer_cutting_distance { {M3 3 4} {M4 4 5} }
prompt> **set_repeater_group_constraints -type placement -horizontal_repeater_spacing 4**

SEE ALSO

add_group_repeaters(2)
place_group_repeaters(2)
remove_repeater_group_constraints(2)
report_repeater_group_constraints(2)

set_report_configuration

Specifies content and format information for a report command.

SYNTAX

status **set_report_configuration**
-report *report_name*
[-columns *column_specs*]
[-filler_string *filler*]
[-overflow_method allow | truncate_left | truncate_middle | truncate_right]
[-dynamic_width *dynamic_flag*]
[-default]

Data Types

report_name string
column_specs list
filler string
dynamic_flag Boolean

ARGUMENTS

-report *report_name*

Specifies the name of the report to configure. Only certain reports are configurable by this command; in particular reports that accept a **-columns** option.

-columns *column_specs*

Specifies the columns to include in the report, their order, and optionally column width and number of significant digits for each column. Each entry is a column spec, which can take the following forms. If just a column name is specified, such as "area", it will use report-specific default column width and significant digits. A column spec such as "area:10" means to use a column width of 10. A column spec such as "area:10.4" means to use a column width of 10 and 4 significant digits. The significant digits information only applies to floating point values. If any column spec is "all", all valid columns will be included for this report.

-overflow_method allow | truncate_left | truncate_middle | truncate_right

Specifies behavior when a value needs more characters than the column width. The default is **allow**, which means to print the value as is and displace values in following columns for that row. Specifying one of the truncation approaches will cause the value to be shortened by truncating characters at the left, middle or right of the value and printing a single "*" character in place of the missing characters. For example, "abcdefghi" will be printed as "abcde*" if the column width is 6 and the overflow method is **truncate_right**. Note that numeric values are never truncated (except by the number of significant digits) because doing so would result in misleading information.

-filler_string *filler*

Specifies a string to be printed in place of empty values. By default, if the value is empty, the value will be printed as blank spaces to fill the column width. Specifying a different string such as "##" for the filler can make the report easier to parse by scripts.

-dynamic_width ***dynamic_flag***

Specifies whether to dynamically increase column widths to fit contents. If true (the default), any column that does not have a user-specified width will be at least as many characters wide as the widest value string for that column. In that case, the column values will not be truncated. If false, columns that do not have user-specified width will have a default width, and values larger than the column width will be truncated unless the overflow_method is **allow**. Dynamic width is usually preferred, unless the overall row width needs to be limited. Note that a report with many rows in a single table will run slightly slower with dynamic width set to true.

-default

Restores the configuration of this report to a default configuration.

DESCRIPTION

This command modifies the format of certain reports, by specifying which columns to show, the order of the columns, the column width, the number of significant digits for the column (for floating point values), a filler string to show instead of empty values, and the behavior in case the value is too large for its column.

EXAMPLES

The following example specifies the configuration to use for **report_lib_cells**. The "full_name" column will be displayed with a width of 10, the "area" column will be displayed with a width of 10 and 4 significant digits, and the "pin_count" column will be displayed with a default width.

```
prompt> set_report_configuration -report report_lib_cells \
 -columns {full_name:20 area:10.4 pin_count} -overflow_method truncate_left
```

SEE ALSO

report_lib_cells(2)

set_retention

Defines the UPF retention strategy for the power domains in the design.

SYNTAX

status **set_retention**
retention_strategy
-domain *power_domain*
[-retention_power_net *retention_power_net*]
[-retention_ground_net *retention_ground_net*]
[-retention_supply *retention_supply_set*]
[-no_retention]
[-elements *objects*]
[-save_signal {*save_signal save_sense*}]
[-restore_signal {*restore_signal restore_sense*}]
[-save_condition *{boolean_function}*]
[-restore_condition *{boolean_function}*]
[-retention_condition *{boolean_function}*]
[-update]

Data Types

retention_strategy string
power_domain string
retention_power_net string
retention_ground_net string
retention_supply_set string
objects list
save_signal string
save_sense high|low
restore_signal string
restore_sense high|low
boolean_function string

ARGUMENTS

retention_strategy

Specifies the name of the UPF retention strategy. The name of the retention strategy must be unique within the specified power domain.

-domain *power_domain*

Specifies the name of the power domain to which this UPF retention strategy should be applied.

-retention_power_net ***retention_power_net***

Specifies the retention power net for the retention cells inferred by this UPF retention strategy.

-retention_ground_net ***retention_ground_net***

Specifies the retention ground net for the retention cells inferred by this UPF retention strategy.

-retention_supply ***retention_supply_set***

Specifies the supply set whose power and ground function associations to be used as the retention power and retention ground nets respectively. This argument is mutually exclusive with **-retention_power_net** and **-retention_ground_net** options.

-no_retention

Specifies that the specified objects in the power domain have no retention.

-elements ***objects***

Specifies the objects to which this UPF retention strategy applies. The objects can be hierarchical cells, leaf cells, HDL blocks, and nets.

-save_signal {*save_signal save_sense*}

Specifies the save signal and the save signal sense for the retention cells under this retention strategy.

-restore_signal {*restore_signal restore_sense*}

Specifies the restore signal and the restore signal sense for the retention cells under this retention strategy.

-save_condition ***{boolean_function}***

Controls the saving of the current register's value in the retention register's shadow latch.

-restore_condition ***{boolean_function}***

Controls the restoration of the value from the register's shadow latch to the register.

-retention_condition ***{boolean_function}***

Controls the preservation of the saved value in the shadow register.

-update

Adds the specified elements to an existing retention strategy. It is an error if the strategy specified does not exist.

-use_retention_as_primary

Specifies that the retention supply powers the output driver of the retention register, and the driver supply of the data output of the retention register is therefore the retention supply.

DESCRIPTION

If you use the **-no_retention** option, the storage elements specified by this strategy do not have retention. This is useful when you want to exclude descendant cells from a different retention strategy that is applied to a hierarchical cell.

If the **-elements** option is not specified, the retention strategy is applied to all the sequential cells in the power domain. If the **-elements** is specified, the UPF retention strategy is applied to all the sequential cells that are under the objects from **-elements**.

If the **-retention_supply** option is not specified and neither the **-retention_power_net**, nor the **-retention_ground_net** options are specified, the retention power and ground nets are automatically set to the power and ground functions of the **default_retention** supply set handle of the power domain for which the strategy is being defined.

If the **-use_retention_as_primary** option is specified with either **-no_retention** option or **-update** option, it is an error.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to define a UPF retention strategy in the specified power domain PD1. Power domain PD1 is defined on instance shutdown_inst.

```
prompt> set_retention retention_1\
-domain PD1\
-retention_power_net PN1 B\
-retention_ground_net GN1
```

The following example shows how to define a UPF retention strategy in the objects under the specified power domain:

```
prompt> set_retention retention_2\
-domain PD1\
-retention_power_net PN1\
-retention_ground_net GN1\
-elements shutdown_inst/mid_inst
```

The following example shows how to define a UPF retention strategy that excludes register R_reg from retention of its parent cell:

```
prompt> set_retention retention_3\
-domain PD1\
-no_retention\
-elements shutdown_inst/mid_inst/R_reg
```

The following example shows how to define a UPF retention strategy using a supply set:

```
prompt> set_retention retention_2 -domain PD1 \
     -retention_supply ret_supply_set \
     -elements shutdown_inst/mid_inst/R_reg
```

The following example shows how a UPF retention strategy can be defined without using a supply set or supply nets.

```
prompt> set_retention retention_2 -domain PD1 \
     -elements shutdown_inst/mid_inst/R_reg
```

SEE ALSO

set_retention_control(2)
map_retention_cell(2)

set_retention_control

Defines the UPF retention control signals for the defined UPF retention strategy.

SYNTAX

status **set_retention_control**
retention_strategy
-domain *power_domain*
-save_signal {*save_signal save_sense*}
-restore_signal {*restore_signal restore_sense*}
[-assert_r_mutex {*net_name sense*}]
[-assert_s_mutex {*net_name sense*}]
[-assert_rs_mutex {*net_name sense*}]

Data Types

retention_strategy	string
power_domain	string
save_signal	string
save_sense	high|low
restore_signal	string
restore_sense	high|low
net_name	string
sense	high|low

ARGUMENTS

retention_strategy

Specifies the UPF retention strategy name. The retention strategy should have already been defined using **set_retention** command

-domain *power_domain*

Specifies the power domain where this UPF retention strategy is applied.

-save_signal {*save_signal save_sense*}

Specifies the save signal and the save signal sense for the retention cells under this retention strategy.

-restore_signal {*restore_signal restore_sense*}

Specifies the restore signal and the restore signal sense for the retention cells under this retention strategy.

-assert_r_mutex {*net_name sense*}

Creates an assertion, which the verification tools can trigger, when the specified RTL signal is active simultaneously with the restore signal.

-assert_s_mutex {*net_name sense*}

Creates an assertion, which the verification tools can trigger, when the specified RTL signal is active simultaneously with the save signal.

-assert_rs_mutex {*net_name sense*}

Creates an assertion, which the verification tools can trigger, when the specified RTL signal is active simultaneously with the save or restore signals. If the signal name and value are not specified, it indicates the save and restore signals are mutually exclusive.

DESCRIPTION

This command defines the UPF retention control signals and control signal sense for the retention strategy. The specified control signals are applied to the retention cells of the retention strategy.

The **-assert_r_mutex**, **-assert_s_mutex**, and **-assert_rs_mutex** options create assertions, which can be used as triggers by the verification and simulation tools.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to define the retention control signals on the defined UPF retention strategy *retention_1* of the power domain *PD1*.

```
prompt> set_retention_control retention_1 \
-domain PD1 \
-save_signal {save_net high} \
-restore_signal {restore_net low}
```

The following example shows how to define the assertion options for the retention strategy retention_2 of the power domain PD1.

```
prompt> set_retention_control retention_2 \
    -domain PD1 \
    -save_signal {sleep high} \
    -restore_signal {wake_up low} \
    -assert_r_mutex {wake_up high} \
    -assert_s_mutex {sleep low} \
    -assert_rs_mutex {wake_up low}
```

The following example shows how to define the assertion options when the save signal and restore signal are mutually exclusive.

```
prompt> set_retention_control retention_3 \
    -domain PD1 \
    -save_signal {sleep high} \
    -restore_signal {wake_up low} \
```

-assert_rs_mutex {}

SEE ALSO

set_retention(2)
map_retention_cell(2)

set_retention_elements

Create a named list of elements whose collective state shall be maintained if retention is applied to any of the elements in the list.

SYNTAX

status **set_retention_elements**
retention_list_name
-elements *objects*

Data Types

retention_list_name	string
objects	list

ARGUMENTS

retention_list_name

Specifies the name of the retention element list.

-elements *objects*

Specifies the objects as part of the retention element list.

DESCRIPTION

The set_retention_elements command defines a list of state elements whose collective state shall be maintained coherently if retention is applied to any of these elements.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows a retention element list is defined with elements "r1" and "r2".

prompt> **set_retention_elements ret_list -elements {r1 r2}**

SEE ALSO

set_retention(2)

set_route_opt_target_endpoints

Assigns rules to determine the targeted endpoint optimization guidelines for the **route_opt** command.

SYNTAX

integer **set_route_opt_target_endpoints**
[-setup_endpoints *file*]
[-setup_timing *file*]
[-hold_endpoints *file*]
[-hold_timing *file*]
[-ldrc_objects *file*]
[-setup_endpoints_collection *pin_list*]
[-hold_endpoints_collection *pin_list*]
[-ldrc_objects_collection *pin_list*]
[-scenario *mode*]
[-reset]

Data Types

file string
pin_list list of pins/ports
mode string

ARGUMENTS

-setup_endpoints *file*

Specifies the name of a file which includes list of endpoints to be considered for setup optimization.

-setup_endpoints_collection *pin_list*

Specifies the list of endpoint pins/ports to be considered for setup optimization.

-setup_timing *file*

Specifies the name of a file which includes list of endpoints and their relevant timing constraints for which max timing adjustments must be considered during the setup optimization.

-hold_endpoints *file*

Specifies the name of a file which includes list of endpoints to be considered for hold optimization.

-hold_endpoints_collection *pin_list*

Specifies the list of endpoint pins/ports to be considered for hold optimization.

-hold_timing *file*

Specifies the name of a file which includes list of endpoints and their relevant timing constraints for which min timing adjustments must be considered during hold optimization.

-ldrc_objects *file*

Specifies the name of a file which includes list of driver terms to be considered for LDRC optimization.

-ldrc_objects_collection *pin_list*

Specifies the list of driver terms to be considered for LDRC optimization.

-scenario *mode*

Specifies the name of a scenario which will be used as the default scenario during application of timing adjustment rules.

-reset

Clears all the targeted endpoint constraints, including marking of the endpoint and/or driver terms for target endpoint optimization, and all min/max timing adjustments on all modified endpoints.

DESCRIPTION

This command reads input files and mark certain endpoints or driver terms to be considered for targeted endpoint optimization using **route_opt** command. User can define these endpoints for setup, and/or, hold, and/or LDRC optimizations. The **route_opt** command will determine which endpoints must be optimized for any of setup, hold and LDRC qualities.

The min/max timing adjustment information on certain endpoints can also be defined by using this command. This can be useful for fixing timing correlations.

This command must be defined prior to the **route_opt** command in order to have proper effect.

NOTE: The intended usage of the targeted optimization flow is to fix very few remaining violations after successfully executing at least two full **route_opt** calls.

Please make sure the bulk of all violations (setup/hold/ldrc) have already been addressed and fixed through executing default flow and by using full **route_opt** command(s).

Unlimited number of violations (for each setup/hold/ldrc) are allowed to get passed to this command. But, Targeted flow will enabled from third route_opt call onwards. That is, user should run route_opt two times prior to running targeted route_opt flow. Please make sure the list of the passed objects to the command are reviewed carefully and only pass those targeted objects that are necessary to be optimized.

EXAMPLES

The following example, reads a sample file 'test.txt' which contains a list of targeted endpoints for setup optimization. Using this setting, the **route_opt** command only performs setup optimization on those specific endpoints listed in 'test.txt' file.

prompt> **set_route_opt_target_endpoints -setup_endpoints test.txt**

SEE ALSO

route_opt(2)

set_routing_rule

Assigns a non-default or default routing rule to specific nets.

SYNTAX

integer **set_routing_rule**
[-rule *rule* | -default_rule | -no_rule]
[-min_routing_layer *layer*]
[-max_routing_layer *layer*]
[-min_layer_mode *mode*]
[-max_layer_mode *mode*]
[-min_layer_mode_soft_cost *cost*]
[-max_layer_mode_soft_cost *cost*]
[-min_layer_is_user true|false]
[-max_layer_is_user true|false]
[-clear]
[*net_list*]

Data Types

rule	string or collection
layer	list or collection
mode	string
cost	string
net_list	list or collection

ARGUMENTS

-rule *rule*

Specifies the name or collection of the non-default routing rule to be assigned to the nets.

The -rule, -default_rule, and -no_rule options are mutually exclusive; you can specify only one.

-default_rule

Specifies that the default routing rule should be assigned to the nets.

The -rule, -default_rule, and -no_rule options are mutually exclusive; you can specify only one.

-no_rule

Removes the current routing rule, if any, from the nets. This allows the tool to automatically assign a non-default rule or the default rule to the nets.

The -rule, -default_rule, and -no_rule options are mutually exclusive; you can specify only one.

-min_routing_layer *layer*

Specifies the minimum routing layer for the nets. Only one layer can be specified.

-max_routing_layer *layer*

Specifies the maximum routing layer for the nets. Only one layer can be specified.

-min_layer_mode *mode*

Specifies the minimum routing layer mode for the nets. Valid values are unknown, extract_only, soft, allow_pin_connection, and hard. The default value is unknown.

-max_layer_mode *mode*

Specifies the maximum routing layer mode for the nets. Valid values are unknown, extract_only, soft, allow_pin_connection, and hard. The default value is unknown.

-min_layer_mode_soft_cost *cost*

Specifies the minimum routing layer mode soft cost for the nets. Valid values are unknown, low, medium, and high. The default value is unknown.

-max_layer_mode_soft_cost *cost*

Specifies the maximum routing layer mode soft cost for the nets. Valid values are unknown, low, medium, and high. The default value is unknown.

-min_layer_is_user true|false

Indicates whether or not the minimum routing layer is user-generated. True indicates it is user-generated. False indicates it is tool-generated. Default value is true.

-max_layer_is_user true|false

Indicates whether or not the maximum routing layer is user-generated. True indicates it is user-generated. False indicates it is tool-generated. Default value is true.

-clear

Clears the nets' values. This resets the nets' routing rule, min/max routing layers, min/max layer modes, and min/max soft costs back to their default values. If this option is specified along with other options, the nets' values are cleared first and then set to the new values specified by the other options.

net_list

Lists of nets to which a routing rule is assigned.

DESCRIPTION

This command assigns a non-default rule, default routing rule, or no rule to specific nets. This command also assigns minimum and maximum routing layer, mode, and cost constraints to the nets.

Non-default rules are defined by using the **create_routing_rule** command and can have associated user-defined width and spacing rules.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example, assigns a non-default rule named *WideMetal* to a net named *B_CLK*.

prompt> **set_routing_rule {B_CLK} -rule WideMetal**

SEE ALSO

create_routing_rule(2)
remove_routing_rules(2)
report_routing_rules(2)

set_rp_group_options

Sets relative placement constraints on the specified relative placement groups.

SYNTAX

int **set_rp_group_options**
rp_group_list
[-tiling_type bit_slice | horizontal_compression | vertical_compression]
[-group_orientation R0 | R180 | MX | MY]
[-alignment left | right | pin [-pin_name *pin_name*]]
[-anchor_corner bottom_left | bottom_right | top_right | top_left | rp_location]
[-x_offset *offset*]
[-y_offset *offset*]
[-anchor_row *row_number*]
[-anchor_column *column_number*]
[-move_effort low | medium | high]
[-optimization_restriction size_only | size_in_place | no_opt | all_opt]
[-rp_only_keepout_margin {*left bottom right top*}]
[-place_around_fixed_cells standard | physical_only | none | all]
[-utilization *utilization_value*]
[-allow_non_rp_cells]
[-allow_non_rp_cells_on_blockages]

Data Types

rp_group_list list or collection
pin_name string
offset float
left float
bottom float
right float
top float
utilization_value double
row_number integer
column_number integer

ARGUMENTS

rp_group_list

Specifies the list of names or collection of relative placement groups, to which the given relative placement constraints will be applied.

-tiling_type bit_slice | **horizontal_compression** | **vertical_compression**

Specifies how to place cells in relative placement groups. The default is bit_slice.

The *bit_slice* value specifies that both row and column alignments are preserved. The next row starts after the tallest object in the current row and the next column starts after the widest object in the current column

The *vertical_compression* value specifies that only column alignment must be preserved. The columns are compressed such that no gap exists between leaf cells, lower-level hierarchical relative placement groups, and blockages in a column.

The *horizontal_compression* value specifies that only row alignment must be preserved. The rows are compressed such that no gap exists between leaf cells, lower-level hierarchical relative placement groups, and blockages in a row.

This option does not propagate to child relative placement groups.

-group_orientation R0 | R180 | MX | MY

Specifies the orientation of a relative placement group. This option does not propagate to child relative placement groups.

-alignment left | right | pin

Specifies the alignment method to use when placing leaf cells and relative placement groups in the specified relative placement groups. If you do not specify this option, the tool uses left alignment. This option does not propagate to child relative placement groups.

-pin_name *pin_name*

Specifies the default alignment pin for the specified relative placement groups. During placement, the tool uses the location of pin to align the cells within a column when you use the pin alignment type. If via regions are present in a pin, the tool tries to align via regions.

You can override the alignment pin for a specific leaf cell when you add the cell to the group with the **add_to_rp_group** command.

-anchor_corner bottom_left | bottom_right | top_right | top_left | rp_location

Specifies the corner for the anchor point that is set by using the **-x_offset** and **-y_offset** options. The valid *corner* values are *bottom_left*, *bottom_right*, *top_left* or *top_right*.

If you specify *bottom_left* (the default), the anchor point corner is the lower-left corner of the relative placement group.

If you specify *bottom_right*, the anchor point corner is the lower-right corner of the relative placement group.

If you specify *top_left*, the anchor point corner is the upper- left corner of the relative placement group.

If you specify *top_right*, the anchor point corner is the upper- right corner of the relative placement group.

If you specify *rp_location*, the anchor point corner is the starting location of object at the row and column specified by the *-anchor_row* and *-anchor_column* options. The options *-anchor_row* and *-anchor_col* are required and the specified location must not be empty.

This option applies to top-level relative placement groups only and is ignored for hierarchical relative placement groups.

A slight deviation from the specified anchor point might occur for non *bottom_left* anchor corners, whenever there is some change in the size of the relative placement group because of optimization or new fixed objects in the vicinity.

-x_offset *offset*

Specifies the x-coordinate of the anchor_corner of the group, in microns, relative to the lower-left corner of the chip area. If you specify only an x-offset, the tool can slide the rp group in the y-direction.

-y_offset *offset*

Specifies the y-coordinate of the anchor_corner of the group, in microns, relative to the lower-left corner in the chip area. If you

specify only a y-offset, the tool can slide the rp group in the x-direction.

-anchor_row ***row_number***

Specifies the row of the object for which the **-anchor_corner rp_location** option is specified.

Note that this option is applicable only to the **-anchor_corner rp_location** option and is not accepted for other values of the **-anchor_corner** option.

-anchor_column ***integer***

Specifies the column of the object for which the **-anchor_corner rp_location** option is specified.

Note that this option is applicable only to the **-anchor_corner rp_location** and is not accepted for other values of the **-anchor_corner** option.

-move_effort low | **medium** | **high**

Controls the extent to which a relative placement group can be moved from its initial location to preserve the relative placement without violating relative placement constraints. **-move_effort high** examines a larger area for possible placement, and **-move_effort low** examines a smaller area.

Coarse placement estimates an initial location for every top-level relative placement group. The relative placement groups may break, if those groups are placed at the same locations returned by coarse placement, which would violate the relative placement constraints for those groups.

Specify **-move_effort high** to allow a larger displacement of a relative placement group from its initial location. The high effort maintains the relative placement constraints and potentially avoids breaking the relative placement group. The trade-off for maintaining the relative placement group is reduced QoR, because the group is moved from the location initially determined by coarse placement.

Specify **-move_effort low** effort to place the relative placement group as close as possible to the initial location returned by coarse placement. The trade-off for maintaining the location is a higher potential for breaking the relative placement groups and violating the relative placement constraints for some relative placement groups.

By default, the tool uses *medium* effort.

-optimization_restriction size_only | **size_in_place** | **no_opt** | **all_opt**

Specifies how to treat the cells in the relative placement group during optimization.

If you specify *size_only*, the cells in the relative placement group can only be sized.

If you specify *size_in_place*, the cells in the relative placement group can only be sized in place.

If you specify *no_opt*, no any optimization will be done on relative placement cells.

If you specify *all_opt*, all the optimization capabilities can be applied to cells in the relative placement group. This might result in losing some cells from the relative placement group because cells can be decomposed or deleted from the netlist.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. The top-level value is propagated to the hierarchical relative placement groups.

Please note that only down sizing is done during *route_opt* for optimization restrictions *size_only*, *size_in_place* and *all_opt*.

By default, the cells in the relative placement group are treated as *size_only* and can only be sized.

-rp_only_keepout_margin {*left bottom right top*}

Specifies the padding margin to be maintained outside of the relative placement group in microns. The specified margin is restricted for other relative placement groups only. The non relative placement cells or the cells of failed relative placement groups can be placed over this margin. If specified margin cannot be respected then relative placement group will be placed

without margin.

The margin parameters must be positive and all four parameters are required.

-place_around_fixed_cells standard | physical_only | none | all

Specifies the legalization strategy for placing relative placement groups.

If you specify *standard*, the relative placement groups are legalized around fixed standard cells and the legalization engine avoids fixed physical-only cells.

If you specify *physical_only*, the relative placement groups are legalized around fixed physical-only cells, and the legalization engine avoids fixed standard cells.

If you specify *none*, the legalization engine avoids both fixed standard and fixed physical-only cells during legalization of relative placement groups.

If you specify *all*, the relative placement groups are legalized around both fixed standard and fixed physical-only cells.

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups.

By default, relative placement groups are legalized around both fixed standard and fixed physical-only cells.

-utilization *utilization_value*

Specifies the utilization percentage for placement of relative placement group. The empty space for meeting utilization is spread throughout the relative placement group after each column. The specified utilization should be between 0.1 and 1.0. The default value for utilization is 1.0, representing 100 percent utilization.

-allow_non_rp_cells

Specifies to allow non relative placement cells in the unused spaces of relative placement group during coarse placement and refine placement. Please note that non relative placement cells are allowed in unused spaces during legalization and optimization.

-allow_non_rp_cells_on_blockages

Specifies to allow non relative placement cells to overlap with relative placement group blockages during coarse placement,refine placement, legalization and optimization.Please note that non relative placement cells are allowed in unused spaces of relative placement group too.

This option has higher priority than **allow_non_rp_cells when both are enabled to set on the relative placement group.**

DESCRIPTION

The **set_rp_group_options** command sets the placement constraints of the specified relative placement groups.

EXAMPLES

The following example uses the **set_rp_group_options** command to change the constraint for a relative placement group.

```
prompt> set_rp_group_options grp_ripple -tiling_type bit_slice
1
```

```
prompt> set_rp_group_options grp_ripple -utilization 0.5
1

prompt> set_rp_group_options grp_ripple \
  -rp_only_keepout_margin {.64 .64 .64 .64}
1

prompt> set_rp_group_options grp_ripple -anchor_corner rp_location \
  -anchor_row 1 -anchor_column 2 -x_offset 100.0 -y_offset 100.0
1
```

SEE ALSO

add_to_rp_group(2)
create_rp_group(2)
get_rp_groups(2)
remove_from_rp_group(2)
remove_rp_group_options(2)
remove_rp_groups(2)
write_rp_groups(2)

set_safety_logic_port_map

Defines the port mapping for the module or lib-cell used for voting logic in functional safety flow.

SYNTAX

status **set_safety_logic_port_map**
[-module *module_list*]
[-mapping *lib_cell_list*]
-input *names*
[-voting *name*]
[-error *name*]

Data Types

module_list	collection
lib_cell_list	collection
names	string_list
name	string

ARGUMENTS

-module *module_list*

Specifies one or more the modules for which the port mapping is applied. Either objects or names can be specified.

-mapping *lib_cell_list*

Specifies one or more libcells for which the port mapping is applied. Either objects or names can be specified.

-input *names*

Specifies list of input ports. This is a compulsory option.

-voting *name*

Specifies the voting logic output. This is optional since some safety logic might not have output port.

-error *name*

Specifies the error port. This is optional since some safety logic might not have error port.

DESCRIPTION

This command marks the different ports of the given module or libcells as inputs, voting outputs or error ports. If different modules and libcells have same port names with same functionality, then multiple such modules and/or libcells can be specified together in the same command invocation. Either -module or -mapping must be specified. Either -voting or -error must be specified.

EXAMPLES

The following example defines a safety logic port map for a lib-cell

prompt> **set_safety_logic_port_map -mapping MAJ3 -input { A B C } -voting { OUT }**

The following example defines a safety logic port map for a module

prompt> **set_safety_logic_port_map -module DMR_LOGIC -input { A1 A2 } -error E**

SEE ALSO

report_safety_logic_port_map(2)
create_safety_register_rule(2)
write_safety_register_script(2)

set_safety_register_rule

Associates the registers with the rules.

SYNTAX

status **set_safety_register_rule**
-rule *safety_register_rule*
-registers *registers*
-error_signal *pin_or_port*

Data Types

safety_register_rule object
registers collection
pin_or_port object

ARGUMENTS

-rule *safety_register_rule*

Specifies the safety_register_rule to apply to the registers. This must an existing and valid rule object or rule name.

-registers *registers*

Explicitly specifies the register(s) on which the constraint should be applied.

-error_signal *pin_or_port*

An existing pin or port in the design to which the error signal generated from voting logic should be tied. This option is compulsory for rules of type dual_mode, otherwise it is optional. It can not be specified for rules of type fault_tolerant. Either a hierarchical or leaf-level pin or an output port can be specified, but it must exist in the design.

DESCRIPTION

Sets the safety_register_rule on all the specified registers. If a specified register already has another rule associated with it, that register will be skipped and the original rule will remain applied to it.

EXAMPLES

The following example creates an association of a rule with registers.

```
prompt> set_safety_register_rule -rule rule1 \
   -registers {flop1 flop2 flop3 flop4}
```

The following example associates a dual_mode rule to registers and specifies an error port

```
prompt> set_safety_register_rule -rule dmr1 \
   -registers {flopA flopB} -error_signal [get_ports Err_out]
```

SEE ALSO

create_safety_register_rule(2)
get_safety_register_rules(2)
write_safety_register_script(2)

set_scaling_lib_group

Specifies the scaling library groups (previously defined by the **define_scaling_lib_group** command) to use for a particular corner of the current design.

SYNTAX

status **set_scaling_lib_group**
[*group*]
[-min]
[-max]
[-all]
[-none]
[-corners *corner_list*]

Data Types

group	string
corner_list	collection

ARGUMENTS

group

Specifies a scaling library group that may be used for maximum and/or minimum delay analysis. It is an error if there is no defined scaling group with the specified name.

-min

Specifies that the specified scaling library group be used for minimum delay analysis. If neither of the **-min** or **-max** options is specified, the scaling library group applies to both minimum and maximum delay analysis.

-max

Specifies that the specified scaling library group be used for maximum delay analysis. If neither of the **-min** or **-max** options is specified, the scaling library group applies to both minimum and maximum delay analysis.

-all

Allow all defined scaling groups to be used in the specified corner(s), for both minimum and maximum delay analysis. This option cannot be used with any other option besides **-corners**.

-none

Prevent any defined scaling groups from being used in the specified corner(s). This option cannot be used with any other option besides **-corners**.

-corners *corner_list*

The corner(s) to apply the scaling group(s) to. If this option is not given, the current corner is used.

DESCRIPTION

The **set_scaling_lib_group** command allows specific scaling library groups to be used for analysis in the specified corner(s). The tool performs interpolation between libraries in this group for voltage, temperature, or both voltage and temperature. You can set scaling library for maximum, minimum, or both maximum and minimum delay analysis. You can set multiple allowable scaling library groups on a corner. By default, all defined scaling groups may be used on every corner.

Multicorner-Multimode Support

EXAMPLES

The following example defines a scaling library group named g1 and applies it to corner C3, for both minimum and maximum delay analysis.

```
prompt> define_scaling_lib_group -name g1 { 0.9.db 1.1.db 1.3.db }
prompt> set_scaling_lib_group -corners C3 g1
```

The following example shows how to prevent any scaling groups from being used for analysis.

```
prompt> set_scaling_lib_group -none -corners [all_corners]
```

SEE ALSO

define_scaling_lib_group(2)
remove_scaling_lib_group(2)

set_scan_compression_configuration

Specifies the scan compression configuration for the design.

SYNTAX

status **set_scan_compression_configuration**
[-inputs *number_of_inputs*]
[-outputs *number_of_outputs*]
[-shared_inputs *number_of_shared_inputs*]
[-shared_outputs *number_of_shared_outputs*]
[-shared_codec_controls false | true]
[-chain_count *chain_count*
| -max_length *max_chain_length*]
[-xtolerance high | default]
[-streaming false | true]
[-serialize false | true]
[-clock *clock_name*]
[-shift_power_groups false | true]
[-shift_power_chain_length *chain_length*]
| -shift_power_chain_ratio *chain_ratio*]
[-shift_power_clock *clock_name*]
[-shift_power_disable *test_control_name*]
[-test_mode *mode_name*]

Data Types

number_of_inputs integer
number_of_outputs integer
number_of_shared_inputs integer
number_of_shared_outputs integer
chain_count integer
max_chain_length integer
mode_name string
chain_length integer
chain_ratio integer
clock_name string
test_control_name string

ARGUMENTS

-inputs

Specifies the number of scan inputs to load scan data for scan compression.

-outputs

Specifies the number of scan outputs to observe scan data for scan compression.

-shared_inputs ***number_of_shared_inputs***

Specifies the number of top-level scan inputs to share across all codecs. The minimum allowed value is the widest input width across all codecs. The maximum allowed value is the sum of all codec input widths, which disables codec I/O sharing. By default, codec I/O sharing is disabled.

-shared_outputs ***number_of_shared_outputs***

Specifies the number of top-level scan outputs to share across all codecs. The minimum allowed value is the widest output width across all codecs. The maximum allowed value is the sum of all codec output widths, which disables output sharing. By default, the outputs are fully shared when codec I/O sharing is enabled by the **-shared_inputs** option.

The **-shared_outputs** option by itself does not enable codec I/O sharing; the feature is only enabled if the **-shared_inputs** option is also specified.

-shared_codec_controls false | true

When set to **true**, the tool adds gating logic at the inputs of the output sharing compressor to selectively disable shared codecs from being observed.

-chain_count

Specifies the number of compressed scan chains to build.

-max_length

Specifies the maximum allowed length of the compressed scan chains.

-xtolerance high | default

Enables the insertion of fully X-tolerant scan compression structures when the value is set to high (the default). When set to default, the tool does not insert fully X-tolerant scan compression structures.

-streaming false | true

When **true**, the specification is a streaming compression configuration. The default is a combinational scan compression configuration.

-serialize false | true

When **true**, the specification is a serialized compression configuration. The default is a combinational scan compression configuration.

-clock ***clock_name***

Specifies the scan clock to be used for the sequential codec (compressor and decompressor) to be inserted. The specified clock must be previously declared as a test clock using the **set_dft_signal** command. By default, the tool selects the dominant scan clock, which typically results in the fewest lockup latches at the decompressor outputs and compressor inputs.

-shift_power_groups false | true

Enables the shift power groups feature. This feature implements a shift power control chain that allows ATPG to selectively gate groups of compressed chains for shift power reduction. The gated chains still shift, but they shift constant (non-toggling) values into the chains, which reduces toggle activity.

When this feature is enabled, the following options are also required:

- **-shift_power_chain_length** or **-shift_power_chain_ratio**

- **-shift_power_clock**

The default is **false**, which does not implement the control chain or gating logic.

-shift_power_chain_length ***chain_length***

Specifies the length of the shift power control (SPC) chain. This value directly controls the number of compressed chain groups.

Use this option to keep the length of the SPC chain constant as the compression architecture changes.

-shift_power_chain_ratio ***chain_ratio***

Specifies the ratio of compressed chains to each shift power control (SPC) register. This value directly controls the number of compressed chains in each group.

Use this option to keep the group size constant as the compression architecture changes.

-shift_power_clock ***clock_name***

Specifies the clock to use for the shift power control chain. The specified clock must be previously declared as a test clock using the **set_dft_signal** command. This option is required when using shift power groups.

-shift_power_disable ***test_control_name***

Specifies the signal that, when asserted to its active value, disables the shift power logic. The specified signal must be previously declared as a TestControl signal using the **set_dft_signal** command. By default, the shift power logic is always active and no shift power disabling logic is implemented.

-test_mode ***mode_name***

Specifies the scan compression test mode to which the specification applies. The default is *all_dft*, except when **define_test_mode** is used. In this case, the default is the value of the last *test_mode* defined.

DESCRIPTION

This command specifies the scan compression configuration for the design.

EXAMPLES

```
prompt> set_scan_compression_configuration -inputs 10 -outputs 10 -chain_count 100
```

SEE ALSO

preview_dft(2)
insert_dft(2)

set_scan_configuration

Specifies the scan configuration for the design.

SYNTAX

status **set_scan_configuration**
[-chain_count *chain_count*
| -max_length *max_chain_length]*
[-clock_mixing no_mix | mix_edges | mix_clocks | mix_clocks_not_edges]
[-add_test_retiming_flops begin_and_end | begin_only | end_only | none]
[-internal_clocks none | single | multi]
[-test_mode *mode_name*]
[-pipeline_scan_enable true | false]
[-pipeline_fanout_limit *max_scan_cells*]
[-insert_terminal_lockup true | false]
[-lockup_type latch | flip_flop]
[-exclude_elements exclude_list]
[-voltage_mixing true | false]
[-power_domain_mixing true | false]

Data Types

chain_count integer
max_chain_length integer
mode_name string
max_scan_cells integer
exclude_list list

ARGUMENTS

-chain_count

Specifies the number of scan chains to build. The tool attempts to meet this constraint. If unable to do so, the tool issues a warning. See the DESCRIPTION section for more information.

-max_length

Specifies the maximum allowed length of the scan chains. The tool attempts to meet this constraint. If unable to do so, the tool issues a warning. See the DESCRIPTION section for more information.

-clock_mixing no_mix | mix_edges | mix_clocks_not_edges | mix_clocks

Specifies whether scan chains can include cells from different clock domains. The following allowed values control what scan elements can be included in the same scan chain:

- **no_mix** (the default)
  Cells must be clocked by the same edge of the same clock.
- **mix_edges**
  Cells must be clocked by the same clock, but the clock edges can be different
- **mix_clocks_not_edges**
  Cells must be clocked by the same clock edge, but the clocks can be different.
- **mix_clocks**
  Cells can be clocked by different clocks and different clock edges.

-add_test_retiming_flops begin_and_end | begin_only | end_only | none

Specifies whether the tool can insert leading edge retiming lockup flip-flops to scan chains clocked with trailing edge scan elements. If a scan chain contains only leading edge scan elements, a retiming lockup flip-flop is not inserted. The following allowed values control the lockup flip-flop insertion during scan insertion:

- **begin_and_end**
  Retiming lockup flip-flops are added to the scan chain beginning and end.
- **begin_only**
  Retiming lockup flip-flop is added only to the scan chain beginning.
- **end_only**
  Retiming lockup flip-flop is added only to the scan chain end.
- **none** (the default)
  No retiming flip-flop is inserted.

-internal_clocks single | none | multi

This option applies only to the multiplexed flip-flop scan style; it is ignored for other scan styles.

The **-internal_clocks** option can be set to the following values:

- **single**
  The **insert_dft** command treats internal clock network regions driven by any combinational gate, including buffers and inverters, as separate clocks for the purposes of scan chain architecture.
- **none**
  The **insert_dft** command does not perform internal clock analysis. This is the default.
- **multi**
  The **insert_dft** command treats internal clock network regions driven by multiple-input gates, such as MUX cells, as separate clocks for the purposes of scan chain architecture. Buffers and inverters are transparent for the analysis.

Integrated clock-gating cells are transparent for the determination of internal clocks.

-test_mode *mode_name*

Indicates which test mode the scan configuration applies to. The default is the last test mode specified with the **current_test_mode** command, or the last test mode created by **define_test_mode** if no current test mode has been set.

-pipeline_scan_enable true | false

When set to **true**, the tool creates a scan-enable pipelining stage for each scan-enable signal. The default is **false**.

-pipeline_fanout_limit *max_scan_cells*

When set to a positive integer value, the tool creates each scan-enable signal so that the number of flip-flops driven by the same scan-enable signal does not exceed the value. This option only takes effect if the **-pipeline_scan_enable** or **-**

domain_based_scan_enable option (or both) is set to **true**. The default is to not apply a limit.

-lockup_type latch | flip_flop

Specifies the kind of cell to use for synchronization between different clock domains. If not specified, the default will be 'latch'.

- **latch**
  Use a level-sensitive latch as clock synchronization element.
- **flip_flop**
  Use an edge-triggered flip-flop as clock synchronization element.

-insert_terminal_lockup

When set to true, inserts synchronization element at the end of scan chains. The default is false.

-exclude_elements

Specifies the list or collection of design objects to exclude from scan chain insertion.

-voltage_mixing true | false

When set to **true**, allows scan elements from different voltage domains into the same scan chain. This requires you to insert level shifters after scan insertion is complete. The default is **false**.

-power_domain_mixing true | false

When set to **true**, allows scan elements from different power domains into the same scan chaini. The default is **false**.

DESCRIPTION

This command specifies the scan configuration for the design.

EXAMPLES

```
prompt> set_scan_configuration -chain_count 10 -max_length 10
```

SEE ALSO

preview_dft(2)
insert_dft(2)

set_scan_element

Sets the **scan_element** attribute on specified design objects, to determine whether scan replacement replaces them with scan cells.

SYNTAX

status **set_scan_element**
true | false
cell_design_ref_list

Data Types

cell_design_ref_list list

ARGUMENTS

true | **false**

The Boolean value with which to set the **scan_element** attribute on the *cell_design_ref_list*. When **true**, (the default), the specified objects are replaced by sequential cells in the design. When **false**, scan replacement is disabled.

cell_design_ref_list

A list of design objects for scan replacement. Objects must be of the following sequential types: - cells (such as flip-flops and latches) - hierarchical cells (containing flip-flops or latches) - references - library cells - designs Wildcards and collections are supported.

DESCRIPTION

Sets the scan_element attribute to true on objects in cell_design_ref_list, indicating that scan replacement is to replace them with equivalent scan cells and make them part of the scan path. This command affects scan replacement performed by compile commands.

The default attribute value is true, which replaces all nonviolated sequential cells with equivalent scan cells for full scan designs. It is only necessary to explicitly set this value to true if you are altering the attribute value of a cell whose attribute was previously set to false.

Nonscan sequential cells determined to be violations by the dft_drc command are not scan-replaced, even if their scan_element attribute is set to true.

Sequential cells with the scan_element attribute set to false are not to be replaced by equivalent scan cells. If the attribute is applied before the first test-ready compile, the cells are never scan-replaced. If the attribute is set to false after a previous test-ready compile, the following behaviors apply:

- Subsequent test-ready compile commands will not unscan them.

For hierarchical cells, the scan_element attribute value applies to all sequential leaf cells within the hierarchical cell. For references, library cells, or designs, the scan_element attribute value applies to the instances of these objects, or to the sequential leaf cells within the hierarchy of the instances.

The effects of the scan_element attribute are applied hierarchically in a design. The value of the scan_element attribute on a lower-level object in the design takes precedence over the value of the scan_element attribute on a higher level object in the design. For example, if the value of the scan_element attribute on a higher-level object in the hierarchy is false and the value of the scan_element attribute on a sequential element inside the higher-level object is true, the sequential element is scan replaced.

The scan_element attribute value on a cell instance takes precedence over the scan_element value on the reference for that cell. The scan_element attribute value on a library cell is treated as a technology limitation and cannot be overwritten in any way.

The set_scan_element command supports full instance-based specification. However, the scan_element attributes on instances are specific to the current design. For example, if you start with a lower-level design, use the set_scan_element command to put scan_element attributes on instances in this context, and change the current design to a higher-level design, the scan_element attributes are no longer visible at the higher level.

To remove individual scan_element attributes, use the **remove_attributes** command. To remove all attributes, use the reset_design command.

To disable scan connections for CTL-modeled custom scan cells that do not use the synthesis-based scan replacement flow, use the set_scan_configuration -exclude command instead.

This command does not affect DFT connections to clock-gating cell test pins; use the set_dft_clock_gating_configuration -exclude_elements command instead.

EXAMPLES

The following example specifies that the compile command is not to scan replace three sequential cells in the current design.

```
prompt> set_scan_element false {U3_1 U3_2 U3_3}
```

The following example specifies that the compile command is not to scan replace instances of library cell DLATCH.

```
prompt> set_scan_element false tech_lib/DLATCH
```

The following example specifies that the compile command is not to scan replace instances of design REGFILE (sequential cells within REGFILE are not to be replaced by equivalent scan cells).

```
prompt> set_scan_element false [get_designs REGFILE]
```

SEE ALSO

preview_dft(2)

set_scan_group

Specifies an unordered group of cells that are not yet connected, but should be kept together within a scan chain. Also identifies existing logic in the current design that is to be designated as a scan segment.

SYNTAX

status **set_scan_group**
scan_group_name
[-access *signal_type_pin_pairs*]
[-include_elements *member_list*]
[-serial_routed false | true]
[-lockup_exists false | true]
[-segment_length *length_of_virtual_segment*]
[-clock *top_level_clock_port*]
[-edge rising | falling]
[-class occ | wrapper | input_wrapper | output_wrapper]

Data Types

scan_group_name string
signal_type_pin_pairs
member_list
length_of_virtual_segment
top_level_clock_port

ARGUMENTS

scan_group_name

Specifies the name of the scan group. The scan group must be uniquely identified as a design object. The name of the scan group name must also be unique.

-access *signal_type_pin_pairs*

Lists ordered pairs, each consisting of a scan signal type and a design pin, indicating how the tool should access the scan group. Valid signal types are ScanClock, ScanMasterClock, ScanSlaveClock, ScanEnable, ScanDataIn, and ScanDataOut. Validation ensures that specified signal types are consistent with the design scan style. Note that this list is mandatory when the **-serial_routed** option is set to **true**. This option is not supported when **-serial_routed** is set to **false**.

-include_elements *member_list*

Specifies an list of scan group components.

If the **-serial_routed** option is set to **false**, the list is unordered, and you can specify sequential cells, segment names, and design

instances.

If the **-serial_routed** option is set to **true**, the list is ordered, and you can specify only sequential cells.

-serial_routed false | **true**

Identifies whether the scan group is composed of serially routed sequential cells or that you are specifying an unordered group of sequential cells. The default is **false**.

-lockup_exists false | **true**

Identifies whether the serially routed scan group has a lock-up latch at the end. The default is **false**.

-segment_length ***length_of_virtual_segment***

Specifies the length of the virtual scan segment. This option can be used only when **-serial_routed** is set to **true**. With this option, you must also use **-access** to specify the correct input and output access pins, and use **-clock** for top-level clock specifications with respect to this segment.

-clock ***top_level_clock_port***

Specifies the top level clock port or pin associated with the virtual scan segment.

-edge rising | **falling**

Specifies the edge of the clock associated with the virtual scan segment. The scan architect uses this information for architecting the scan chains.

-class occ | **wrapper** | **input_wrapper** | **output_wrapper**

Specifies the class of the scan group.

The **occ** keyword specifies that the scan group describes an on-chip clocking (OCC) clock chain segment. Scan groups of **-class occ** are used only during OCC flows.

The **wrapper** keyword specifies that the scan group describes a wrapper chain segment. Scan groups of **-class wrapper** are used only during core wrapping flows.

The **input_wrapper** keyword specifies that the scan group describes an input wrapper chain segment. Scan groups of **-class input_wrapper** are used only during core wrapping flows.

The **output_wrapper** keyword specifies that the scan group describes an output wrapper chain segment. Scan groups of **-class output_wrapper** are used only during core wrapping flows.

DESCRIPTION

The **set_scan_group** command identifies existing logic in the current design that is to be designated as a scan group. Scan groups implement scan in a particular scan style. DFT Compiler can include scan groups in scan chains by connecting their access pins (when serially routed), or maintain the cells' grouping without assuming they are serially connected to each other. The tool does not scan-replace scan group members explicitly.

When the **-serial_routed** option is set to **true**, the **set_scan_group** command allocates sequential cells to scan groups and identifies design pins as scan group access pins with particular scan signal-type semantics.

The **set_scan_group** command options are not incremental. A **set_scan_group** option that specifies a scan group name overwrites any previous **set_scan_group** command specification of the same name and scan style.

The **set_scan_group** command accepts a list of members. By default, the list of members is unordered and not serially connected.

This list must be ordered when **-serial_routed** is set to **true**.

The scan group members should belong to the same clock domain and same edge. If the scan group is provided with elements from a different clock domain, an error message is displayed during scan architecting and the scan group specification is discarded.

The **set_scan_group** command does not accept collections as input.

The **set_scan_group** command accepts design instances as scan group members. It adds every scannable cell in the design instance to the scan group at the specified position, in alphanumeric order. This specification is valid only when **-serial_routed** is set to **false**.

The **-segment_length** option specifies the length of the virtual scan segment. Virtual segments are segments for which you specify the scan segment input and the scan segment output using the **-access** option, and the scan segment clock association using **-clock** option. If you do not specify the **-include_elements** option but you want specify the length of a segment, use the **-segment_length** option. The tool connects only to the hookup points during scan chain construction and does not trace within the hookup points (or virtual scan segment). The segment length you provide is considered for top-level scan chain balancing.

Note that the post-DFT DRC and SCANDEF generation support is not available when virtual scan segments are present in the scan chain.

The **-clock** option specifies the top-level clock that drives the virtual scan segment. If you specify the **-segment_length** option, you must also specify a valid **-clock** specification.

Ensure that the clock name specified as part of the **-clock** option is already defined using the **set_dft_signal** command. Alternatively, run the **set_scan_group** command after creating the test protocol and inferring the clocks and asynchronous signals.

The **-edge** option specifies the controlling edge of the clock with respect to the virtual scan segment. The controlling edge is specified as **rising** or **falling**. The default is **rising**.

When the **-access** option is used, the access pin signal types must be consistent with the segment scan style. Access pin directions must be consistent with their signal types. Access pins cannot be associated with more than one signal type. Scan groups cannot have more than one ScanDataIn or ScanDataOut access pin. Scan groups cannot contain duplicate members; sequential cells cannot belong to more than one scan group. Scan group members cannot also belong to scan chains. In other words, a sequential cell element that belongs to a scan group cannot be specified as part of the **set_scan_path** command.

A scan group does not accept another scan group as a member; nested scan groups are not supported.

Valid scan group error messages are TEST-400 and TEST-700. TEST-400 is displayed when scan group elements belong to more than one clock domain or have the same clock domain but a different edge. TEST-700 is displayed when scan group elements have a nonexistent element specified as part of the scan group specification.

Use the **preview_dft** command to all to review your scan group specifications.

EXAMPLES

The following example identifies an embedded shift register in the multiplexed flip-flop scan style. This specification can be completed in a single **set_scan_group** command by including the identification of the ScanEnable access pin in the first **-access** option specification.

```
prompt> set_scan_group my_shift_reg \
    -access [list ScanDataIn P/B/U7/si ScanDataOut P/B/U9/QN] \
    -include_elements [list P/B/U7 P/B/U8 P/B/U9] \
    -serial_routed true

prompt> set_scan_group my_unconnected_group \
    -include_elements [list U1 U2]
```

The following is an example of a virtual scan segment controlled by the rising edge of the clock tck:

```
prompt> set_scan_group virtual_scan_segment \
      -access [list ScanDataIn P/B/U7/si ScanDataOut P/B/U9/QN] \
      -segment_length 20 \
      -clock tck \
 -edge rising \
      -serial_routed true
```

The following is an example of a virtual scan segment controlled by the falling edge of the clock tck:

```
prompt> set_scan_group virtual_scan_segment \
      -access [list ScanDataIn P/B/U7/si ScanDataOut P/B/U9/QN] \
      -segment_length 20 \
      -clock tck \
 -edge falling \
      -serial_routed true
```

SEE ALSO

preview_dft(2)

set_scan_path

Specifies a scan chain for the current design.

SYNTAX

status **set_scan_path**
chain_name
[-scan_data_in *port_name*]
[-scan_data_out *port_name*]
[-class scan | wrapper | occ | spc | ieee_1838_3dcr]
[-include_elements *element_list*]
[-ordered_elements *element_list*]
[-head_elements *element_list*]
[-tail_elements *element_list*]
[-complete true | false]
[-scan_enable *port_name*]
[-test_mode *mode_name*]
[-input_wrapper_cells_only enable | disable]
[-outputt_wrapper_cells_only enable | disable]
[-scan_master_clock *clock_name*]
[-edge rising | falling]

Data Types

chain_name string
port_name string
element_list list
mode_name string
clock_name string

ARGUMENTS

chain_name

Specifies a name for the scan chain being specified.

-scan_data_in *port_name*

Specifies a scan-in port to be associated with a scan chain. The specified scan_in_port must be previously declared as a ScanDataIn using the **set_dft_signal** command.

-scan_data_out *port_name*

Specifies a scan-out port to be associated with a scan chain. The specified scan_out_port must be previously declared as a ScanDataOut using the **set_dft_signal** command.

-class scan | **wrapper** | **occ** | **spc** | **ieee_1838_3dcr**

Specifies the usage of the current **set_scan_path** command. The allowed values are:

- **scan (default)** specifies that the chain is a scan chain. Chains of class **scan** are used only during scan insertion.
- **wrapper** specifies that the chain is a wrapper chain of a wrapped core. Chains of class **wrapper** are used only during core wrapping insertion.
- **occ** specifies that the chain is a clock chain for an on-chip clocking (OCC) controller. Chains of class **occ** are used only in an OCC flow.
- **spc** specifies that the chain is a shift power control (SPC) chain, used by the shift power groups feature. Chains of class **spc** are used only when the SPC feature is enabled.
- **ieee_1838_3dcr** "specifies that the chain is 3DCR register. This needs IEEE1838 enabled as a prerequisite. An instruction must be defined to target the 3DCR register.

Scan chains defined as a certain class automatically include elements of that class, unless limited or explicitly specified by other options of this command. The exceptions are the **occ** and **spc** classes, which require the scan elements to be explicitly specified if cores containing occ or spc chains are present.

-include_elements ***element_list***

Specifies a list of scan chain elements to be included in the scan chain. Unlike an ordered list, the scan architect is free to place them anywhere in the scan chain. This list can include names of sequential cells, design instances, scan segments, and scan links. The input is accepted as a simple list. Wildcards and collections are not supported.

-ordered_elements ***element_list***

Specifies an ordered list of scan chain elements to be included in the scan chain. This list can include names of sequential cells, design instances, scan segments, and scan links. The input is accepted as a simple list. Wildcards and collections are not supported.

-head_elements ***element_list***

Specifies an ordered list of scan chain elements to be placed at the beginning of the scan chain. This list can include names of sequential cells, design instances, scan segments, and scan links. The input is accepted as a simple list. Wildcards and collections are not supported.

-tail_elements ***element_list***

Specifies an ordered list of scan chain elements to be placed at the end of the scan chain. This list can include names of sequential cells, design instances, scan segments, and scan links. The input is accepted as a simple list. Wildcards and collections are not supported.

-complete true | **false**

Indicates whether the tool can add components to a specified scan chain. When **true**, the tool does not add any other scan cells to this scan chain and treats it as a complete scan chain. When **false** (the default), the tool can add more scan cells to this specification to balance scan chains. The new scan cells are embedded at the beginning of the scan chain (after any **-head_elements** scan cells).

-scan_enable ***port_name***

Specifies a scan enable to be associated with a scan chain. The scan architect takes the specified scan-enable port into consideration while allocating a scan-enable signal to the scan chain. The specified scan-enable port is dedicated to the scan chain; it will not be used to drive scan enable pins of flip-flops in other chains. If the internal pins flow is enabled using the **set_dft_drc_configuration** command, a hookup pin can be specified as a scan-enable signal for a scan chain. The specified scan_enable must be previously declared as a ScanEnable using the **set_dft_signal** command.

-test_mode ***mode_name***

Indicates the test mode to which the scan path constraint applies. The default is the current test mode, if user-defined test modes have been created, or the **Internal_scan** mode if no test modes have been created. Note that unlike most commands, the **set_scan_path** command does not apply a global specification to all test modes when the **-test_mode** option is omitted and no current test mode has been defined. To apply a scan path specification to all test modes, you must explicitly specify the **-test_mode all_dft** option.

-input_wrapper_cells_only enable | **disable**

Specifies whether or not only input wrapper cells should be used in the chain. This option is specific to chains of class **wrapper**. The default for this option is disable.

-output_wrapper_cells_only enable | **disable**

Specifies whether or not only output wrapper cells should be used in the chain. This option is specific to chains of class **wrapper**. The default for this option is disable.

-scan_master_clock ***clock_name***

Specifies a clock to be associated with a scan chain. The scan architect takes the user-defined clock domain into consideration while building scan chains. The clock edge is determined by the **-edge** option, which defaults to rising-edge scan cells. If the same clock edge specification applies to more than one scan chain specification, the scan architect attempts to balance the clock domain elements across the scan chains.

-edge rising | **falling**

Specifies the edge of the scan_master_clock associated with the scan chain. The scan architect includes only the elements controlled by the specified edge of the scan_master_clock in the scan chain. The default is rising.

DESCRIPTION

This command specifies a scan chain for the current design. The command allocates scan cells, scan segments, and scan links to scan chains, and specifies scan chain orderings.

Scan chain elements cannot belong to more than one chain. When **set_scan_path** options conflict, the most recent command is rejected.

The **set_scan_path** options are not incremental. A **set_scan_path** option specified for a given chain name overwrites any previous **set_scan_path** option previously applied to that same chain name.

Scan segments can be subdesign scan chains. Suppose the hierarchical design instance *instance_name* has a scan chain of **report_dft**. The **-scan_path** option reports as "Complete scan chain #n". You can include this subdesign scan chain in a scan chain by referring to it as *instance_name/n*.

Scan chain orderings must satisfy clock domain constraints asserted by using the **set_scan_configuration** command with the **-clock_mixing option**. The tool resynchronizes nonfunctional scan chains by using lockup latches.

The **set_scan_path** command does not invalidate **dft_drc** modeling information.

Use the **report_scan_path** command to review your scan chain specification. Use **compile** or **insert_dft** commands to implement the scan chain.

In the case of multimode scan insertion, if you have previously defined several test modes using the **define_test_mode** command, the default **set_scan_path** command is not associated with any mode and is not honored. Use the **-test_mode** option to specify the test mode to which you want your scan path constraint to apply.

EXAMPLES

The following example defines a complete scan chain named C1, consisting of sequential cells A, B, and C in reverse alphanumeric order.

```
prompt> current_design WC66
{WC66.design}
prompt> set_scan_path C1 -ordered_elements {C B A} -complete true
1
```

The following example shows how to use the **-head_elements** and **-tail_elements** options to specify that the instances U4/1 and U4/2 are at the beginning of the chain and the instances U5/1 and U5/2 are at the end of the chain. The **-include_elements** option ensures that the U2/2 segment is included somewhere in scan chain C1. The instances U3/2 and U3/1 are ordered and scan architect does not change the order of the elements in these instances during architecting.

```
prompt> set_scan_path C1 -view spec -head_elements {U4/1 U4/2} \
      -include_elements {U2/2} -ordered_elements {U3/2 U3/1} \
      -tail_elements {U5/1 U5/2}
1
```

The following example shows how to specify the DFT signals and scan path for a design with existing scan chains:

```
prompt> set_dft_signal -view existing_dft -type ScanDataIn -port test_si
Accepted dft signal specification for all_dft mode
1
prompt> set_dft_signal -view existing_dft -type ScanDataOut -port test_so
Accepted dft signal specification for all_dft mode
1
prompt> set_dft_signal -view existing_dft -type ScanEnable -port test_se
Accepted dft signal specification for all_dft mode
1
prompt> set_scan_path C1 -scan_data_in test_si \
      -scan_data_out test_so -scan_enable test_se
1
```

The following examples show how to use **set_scan_path** with the **-class occ** option. The clock controllers occ_ctrl and occ_ctrl2 have 2 chain_counts.

```
prompt> set_dft_clock_controller -cell_name occ_ctrl \
      -ateclock ate_clk -pllclocks {...} -chain_count 2
1
prompt> set_dft_clock_controller -cell_name occ_ctrl2 \
      -ateclock ate_clk -pllclocks {...} -chain_count 2
1
```

A basic case with 1 clock chain:

```
prompt> set_scan_path C1 -scan_data_in SI10 -scan_data_out SO10 \
      -include_elements {occ_ctrl1 occ_ctrl2} -class occ
1
```

This next example define 2 clock chains. In this case, the clock chains of the clock controllers can be specified using indexes as follows:

```
prompt> set_scan_path C2 -scan_data_in SI10 -scan_data_out SO10 \
```

```
        -include {occ_ctrl:1 occ_ctrl2:0} -class occ
1
prompt> set_scan_path C3 -scan_data_in SI11 -scan_data_out SO11 \
        -include {occ_ctrl:0 occ_ctrl2:1} -class occ
1
```

The following examples show how to use the **-edge** option along with the **-scan_master_clock option**. First, the **set_scan_path** command assigns all elements controlled by the rising edge of the clk1 clock to C1, and then assigns all elements controlled by the falling edge of the clk1 clock to C2.

```
prompt> set_scan_path C1 -scan_master_clock clk1 -edge rising
1
prompt> set_scan_path C2 -scan_master_clock clk1 -edge falling
1
```

Consider **set_scan_path** specifications for C3 and C4 below. If there are 10 elements controlled by the rising edge of clock clk2, then C3 and C4 will get 5 elements each.

```
prompt> set_scan_path C3 -scan_master_clock clk2 -edge rising
1
prompt> set_scan_path C4 -scan_master_clock clk2 -edge rising
1
```

SEE ALSO

insert_dft(2)
preview_dft(2)
report_scan_path(2)
set_dft_clock_controller(2)

set_scan_rp_group

Configures a DFT Scan relative placement group in the given design.

SYNTAX

status **create_rp_group**
rp_group_list
-routing *direction*
[-starting_corner *corner*]
[-max_length *size*]

Data Types

rp_group_list list or collection
direction string
corner string
size positive integer

ARGUMENTS

rp_group_list

Specifies the name, a list or a collection of the relative placement groups.

-routing *direction*

Specifies the direction that the DFT Scan Synthesis tool should use to stitch the RP group cells into Scan chains.

The valid *direction* values are as follows:

- **horizontal** indicates the RP Scan cells must be stitched in an order following the rows of the RP group.
- **vertical** indicates the RP Scan cells must be stitched in an order following the columns of the RP group.

-starting_corner *corner*

Specifies the corner of the RP group where the DFT Scan stitching should start from.

The valid *corner* values are as follows:

- **ll** indicates the RP Scan cells must start stitching RP Group Scan cells from the lower-left corner of the RP group.
- **lr** indicates the RP Scan cells must start stitching RP Group Scan cells from the lower-right corner of the RP group.
- **ul** indicates the RP Scan cells must start stitching RP Group Scan cells from the upper-left corner of the RP group.

- **ur** indicates the RP Scan cells must start stitching RP Group Scan cells from the upper-right corner of the RP group.

The default value for this option Is **ll** (lower-left).

If in the selected corner position of the RP group there is no available Scan cell to be stitched, the next cell available in the direction defined by the -routing option will be selected as the starting cell for DFT Scan stitching.

-max_length *size*

Specifies the desired size of the +ORDERED groups to be created while stitching the RP Scan cells. Default value is 0, and therefore each +ORDERED group will contain all cells of each row or column, depending of the -routing direction selected.

If a positive value is set for -max_length, the RP Group will stitch as many cells as possible in a single +ORDERED group until it reaches the desired max_size or if completes a row/column of cells of the RP group.

If the specified *size* is smaller than the amount of Scan cells of one row/column, at least a complete row/column will be set as an +ORDERED group.

If the specified *size* is greater than the amount of Scan cells of one row/column, one or more than one row/columns will be set into the same +ORDERED group, but always grouping complete rows/columns of RP cells.

Specifying a *size* equal or greater than the value of rows times columns of the RP group will imply all Scan cells of the RP group will be stitched in a single big +ORDERED group, following a stitching order given by the provided *direction* and *corner*, in a snake-like order.

DESCRIPTION

The **set_scan_rp_group** command defines a dedicated DFT Scan stitching style to be applied to specific relative placement (RP) groups previously defined by the **create_rp_group** command.

DFT Scan Synthesis normally can freely choose available Scan cells of the design to architect the Scan chains, including cells belonging to RP groups. This freedom can produce multiple new Scan path routing to ScanInput or ScanOutput, and those tracks will also be added to the RP group; as RP groups cells are very tightly placed, the Scan stitching process can introduce congestion in the RP group while competing with functional Data path scarce routing resources.

Using the new **set_scan_rp_group** command, user can define a preferred Scan stitching direction for the RP group. If set orthogonal to the functional data path routing direction, it helps easing overall routing of the relative placement group. Besides, the RP group cells will be stitching in a strict order following the RP cell location, reducing the Scan stitching net length inside the RP group.

Depending on the specified Scan routing direction, the command will define one or more Scan groups of RP cells, with the cells or the RP groups ordered by their correlative RP position. This will produce multiple Scan groups that will keep their relative order even after DFT Optimizations. For that objective, the Scan groups formed by the rows or columns of RP cells will be set as an +ORDERED component in ScanDEF file.

The +ORDERED components created from the RP group cells will retain their internal stitching RP cell order unchanged during the flow, but the order in which the multiple +ORDERED groups are stitched together is not fixed, and DFT Scan architecting or DFT optimizations (Scan reordering or repartitioning) are free to change their stitching order to reach other Scan requirements like Scan chain length/balancing or Scan chain net length.

The **set_scan_rp_group** command will honor the RP cell group position and the specified Scan routing *direction*, creating +ORDERED groups from the specified starting *corner*, following a snake-like traversal. For example, if routing "horizontal" is selected together with starting corner "ll", the bottom row of the RP group will be stitched as an +ORDERED group from left to right, the row above the bottom will be stitched as an +ORDERED group from right to left, then the next row above will be stitched as an +ORDERED group from left to right, and so on.

Calling **set_scan_rp_group** command over the same RP groups already set will overwrite the previous defined setup. A warning message will be issued.

The expected result of the command will be observed in the DFT Scan stitching process. Using this command after the design was DFT inserted during commands **compile_fusion** or **insert_dft**, and therefore the Scan chains were already stitched, will not change the original stitching outcome.

The Scan RP groups defined by command **set_scan_rp_group** can be reported using command **report_scan_rp_group** or deleted using command **remove_scan_rp_group**. In the last case only the Scan stitching setup is removed, but the relative placement group remains untouched.

EXAMPLES

The following example creates a Scan relative placement setup for RP group named "top_rp", using "vertical" Scan stitching routing direction, starting from the upper right corner.

```
prompt> set_scan_rp_group top_rp -routing horizontal -starting_corner ur
```

The following example creates a Scan relative placement setup for all RP group that match the pattern "*bottom_rp*" using "horizontal" Scan stitching routing direction, starting from the lower left corner (default value).

```
prompt> set_scan_rp_group top_rp -routing vertical [get_rp_groups *bottom_rp*]
```

SEE ALSO

create_rp_group(2)
get_rp_groups(2)
write_scan_def(2)
compile_fusion(2)
insert_dft(2)
report_scan_rp_group(2)
remove_scan_rp_group(2)

set_scan_skew_group

Defines a scan skew group of scan cells, which might have a different clock latency characteristic than other parts of the design.

SYNTAX

status **set_scan_skew_group**
scan_skew_group_name
-include_elements *object_list*

Data Types

scan_skew_group_name string
object_list list

ARGUMENTS

scan_skew_group_name

Specifies the name of the scan skew group to define.

A scan skew group is referenced by name. You can use any name, as long as it has not previously been used to define another scan path or scan skew group. Scan skew group names are used by the **report_scan_skew_group**, **remove_scan_skew_group**, and **preview_dft** commands.

-include_elements *object_list*

Specifies the cells to include in the scan skew group. The object list can contain leaf cells, hierarchical cells, and CTL-modeled cells. Hierarchical cells include all scan cells in their hierarchy. CTL-modeled cells include all their scan segments. Wildcards and collections are supported.

DESCRIPTION

You can use scan skew groups to provide manual guidance for lock-up latch insertion. A scan skew group is a group of scan cells that might have a different clock latency characteristic than other parts of the design. The DFT architect treats the scan skew group as if it were a unique clock domain.

Scan skew groups override the normal scan clock domain identification behaviors such as

- Scan clock name
- The **-internal_clocks** option of the **set_scan_configuration** command

If clock mixing is disabled, the cells in a scan skew group are not mixed in the same scan chain as other scan cells. If clock mixing is enabled, the cells in a scan skew group can be mixed in the same scan chain as other scan cells, with lock-up latches inserted as needed.

EXAMPLES

The following example defines a scan skew group for a particular design register:

```
prompt> set_scan_skew_group G1 -include_elements {Z1P_reg[10] Z1P_reg[11] Z1P_reg[12]}
1
```

SEE ALSO

remove_scan_skew_group(2)
report_scan_skew_group(2)

set_scenario_status

Configures analysis settings for scenarios.

SYNTAX

status **set_scenario_status**
scenario_list
[-setup true | false]
[-hold true | false]
[-leakage_power true | false]
[-dynamic_power true | false]
[-max_transition true | false]
[-max_capacitance true | false]
[-min_capacitance true | false]
[-cell_em true | false]
[-signal_em true | false]
[-active true | false]
[-all | -none]

Data Types

scenario_list list

ARGUMENTS

scenario_list

Specifies the scenario(s) to configure. The command configures each of these scenarios based on other command options. This option must be specified.

-setup true | false

Specifies whether setup analysis is turned on for the specified scenarios.

-hold true | false

Specifies whether hold analysis is turned on for the specified scenarios.

-leakage_power true | false

Specifies whether leakage power analysis is turned on for the specified scenarios.

-dynamic_power true | false

Specifies whether dynamic power analysis is turned on for the specified scenarios.

-max_transition true | **false**

Specifies whether max-transition DRC is turned on for the specified scenarios.

-max_capacitance true | **false**

Specifies whether max-capacitance DRC is turned on for the specified scenarios.

-min_capacitance true | **false**

Specifies whether min-capacitance DRC is turned on for the specified scenarios.

-cell_em true | **false**

Specifies whether cell EM analysis is turned on for the specified scenarios.

-signal_em true | **false**

Specifies whether signal EM analysis is turned on for the specified scenarios.

-active true | **false**

Specifies whether any analysis will be actually done on the scenario. If true, analysis will actually be performed, if any analysis settings are turned on. If false, all analysis settings are temporarily disabled and can be reactivated by specifying **-active true** at a later time.

-all

Turns on all analysis types. The "active" setting will not be affected. If the scenario is inactive, analysis will not be done.

-none

Turns off all analysis types. The "active" setting will not be affected. Only one of the **-all** and **-none** options may be given.

DESCRIPTION

This command enables or disables timing analysis, power analysis, and DRC checking for scenarios in the current design. Setup analysis, hold analysis, power analysis, and various types of DRC and EM checking can be configured independently.

Note that the **create_scenario** command creates a scenario and makes it active for all analysis types except cell and signal EM. The **set_scenario_status** command will generally be needed after a scenario is created to set only the desired analysis types.

The **-all** or **-none** options may be given with the individual analysis type options. In this case, all analysis types will first be set (or cleared), and then the individual analysis options will be applied.

The **-min_capacitance** option also needs **-hold** option to see the min_capacitance violations.

EXAMPLES

The following commands set analysis and DRC options for the func::RCworst and func::RCbest scenarios.

```
prompt> set_scenario_status func::RCworst \
  -setup true -max_capacitance true -max_transition true \
  -min_capacitance false -hold false -leakage_power false \
```

```
    -dynamic_power false -active true

  prompt> set_scenario_status func::RCbest \
    -setup false -max_capacitance true -max_transition true \
    -min_capacitance true -hold true -leakage_power false \
    -dynamic_power false -active true
```

The following command will turn on hold analysis for the scenario, and leave all other analysis settings unchanged.

```
  prompt> set_scenario_status func::RCworst -hold true
```

The following command will ensure that the scenario is configured only for hold analysis, no matter what its previous settings were.

```
  prompt> set_scenario_status func::RCworst -none -hold true -active true
```

The following command will ensure that the scenario is configured for all analysis types except power, no matter what its previous settings were.

```
  prompt> set_scenario_status func::RCworst -all -leakage_power false \
    -dynamic_power false -active true
```

SEE ALSO

create_scenario(2)
report_scenario(2)
remove_scenario(2)

set_scope

Specifies the hierarchical scope of the design for subsequent commands.

SYNTAX

string **set_scope**
[*instance*]

Data Types

instance string

ARGUMENTS

instance

Specifies the working instance on which subsequent commands operate:

- If omitted, the focus moves to (or remains at) the top level of the current design.
- If an instance name, the focus moves down to the named instance in the current level. Use multiple names separated by slash characters to traverse multiple levels down the hierarchy.
- If a single period character ".", the tool stays as the current working instance.
- If two periods "..", the context moves up one level in the hierarchy of the current design.
- If a slash character "/" followed by an instance name, the tool moves to the named working instance in the top level of the design. Use multiple names separated by slash characters to traverse multiple levels down the hierarchy, starting from the top.

DESCRIPTION

The **set_scope** command specifies the hierarchical scope in which subsequent commands operate.

Functionally, this command is similar to the **current_instance** command but gives a different return value. Upon success, the **set_scope** command returns the scope *before* execution of the command as a full path string without a leading slash character, whereas the **current_instance** command returns the name of the current instance *after* execution of the command, with a leading slash character.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command sets the scope to the U1/U8 hierarchical cell. Cell U1 must exist at the current level of hierarchy and cell U8 must exist in the U1 cell.

```
prompt> set_scope U1/U8
```

The following command sets the scope to two levels of hierarchy above the current instance, and then down one level to the U4 cell.

```
prompt> set_scope ../../U4
```

The following script sets the scope to one block and then another, and defines a different power domain in each scope.

```
set_scope U1
create_power_domain PD1
set_scope ..
set_scope U2
create_power_domain PD2
set_scope ..
```

SEE ALSO

current_design(2)
current_instance(2)

set_segment_budget_constraints

Sets a manual constraint for a segment along a budget path.

SYNTAX

int **set_segment_budget_constraints**
-delay *delay*
-remove
[-from *pin*]
[-to *pin*]
[-rule *rule_name*]

Data Types

delay float
pin string
rule_name string

ARGUMENTS

-delay *delay*

Specifies the amount of fixed delay to be allocated to the specified segment of a budgeted path. You must specify either the **-delay** or **-remove** option.

-remove

Removes any previously set segment delay on the specified segment. You must specify either the **-delay** or **-remove** option.

-from *pin*

Specifies a budgeted pin at the start of the segment you want to constrain. If you omit this option, you will constrain the first budget segment from random logic to the given "to" pin. You must specify either **-from** or **-to**, or both options. When they are both specified, the **-from** pin has to be different than the **-to** pin.

-to *pin*

Specify a budgeted pin at the end of the segment you want to constrain. If you omit this option, you will constrain the last budget segment from the given "from" pin to random logic. You must specify either **-from** or **-to**, or both options. When they are both specified, the **-from** pin has to be different than the **-to** pin.

-rule *rule_name*

Specifies the name of a net estimation rule that was used to derive the delay in this constraint. This option is for record keeping only. It does not affect the actual operation of the budgeter. The rule name will be preserved when you save your design. See the **set_net_estimation_rule** man page for more details.

DESCRIPTION

This command sets a manual delay constraint for a segment along a budget path. When the delay is set, the **compute_budget_constraints** command will try very hard to honor the delay constraint. The budget will be allocated such that the delay along the given segment is no more or no less than the delay you have specified. If the path through the selected segment has positive or negative slack, the slack will be allocated to segments along the path that do not have a segment budget constraints.

The budget segment is specified by using the **-from** and **-to** options. The segment delay constraint is applied to the part of the budgeted path between the given pins. It is required that the "from" and "to" pins are two different pins on blocks that were declared as budget blocks by the **set_budget_options** command. You also need to pick pins which are adjacent along a timing path. If the specified "from" and "to" pins are separated by yet another budget pin, then the constraint will be ignored.

You can use the **create_budget_busplan** command and the **-busplans** option of the **compute_budget_constraints** command to define many segments constraints automatically.

EXAMPLES

Constrain the part of a budget path between block1 and block2 to be 1.5.

```
prompt> set_segment_budget_constraints -delay 1.5 \
  -from block1/outpin -to block2/inpin
```

SEE ALSO

compute_budget_constraints(2)
create_budget_busplan(2)
report_budget(2)
set_budget_options(2)
set_net_estimation_rule(2)
write_budgets(2)

set_self_gating_objects

Controls the behavior of self gating insertion for the specified objects. Objects can be of type register, hierarchical cell, power domain, or module.

SYNTAX

status **set_self_gating_objects**
[-include *object_list*]
[-force *object_list*]
[-exclude *object_list*]
[-tree_type *combinational_logic*]
[-clear *object_list*]
[-reset]
[-tree_type *combinational_logic*]

Data Types

object_list list
combinational_logic xor | or | nand | auto

ARGUMENTS

-include *object_list*

Specifies a list of objects in the current design for which self gating should be done as per the options specified by the **set_self_gating_options** command. This is the default for all objects in the design. This option cannot be used in combination with the other ones, except for **-tree_type**.

-force *object_list*

Specifies a list of objects in the current design for which self gating will forcefully be done as options specified by the **set_self_gating_options** command. The specified objects will be self gated even if they don't pass the internal filters. This option will take precedence to the interaction with clock gates setting. This option cannot be used in combination with the other ones, except for **-tree_type**.

-exclude *object_list*

Specifies a list of objects in the current design to be excluded from self gating. This option cannot be used in combination with any other.

-clear *object_list*

Specifies a list of objects in the current design for which the settings previously stored by the command should be removed. This option cannot be used in combination with any other.

-reset

Removes the settings previously stored by the command from all objects in the design. This option cannot be used in combination with any other.

-tree_type *combinational_logic*

Specifies the type of combinational cell which will be used for self gating a register. The available combinational cells are XOR, OR and NAND cells. By default, self gating will use XOR cells. This option should be used in combination with *-include* or with *-force*, and its type specification will be applied to the objects specified by those options.

DESCRIPTION

This command specifies the objects on which self gating is to be either enabled, forced or disabled on its registers, as well as the type of combinational logic that should be used as comparison function.

Use the **-include** option to specify that self gating is allowed on the specified objects, letting the tool decide when the insertion actually happens based on quality of results analysis.

Use the **-force** option to specify that self gating should be forcefully done on the specified objects.

Use the **-exclude** option to specify that self gating should not be done on the specified objects.

Use the **-tree_type** option to specify the type of combinational logic to be used as comparison when self gating the objects specified by the **-include** or the **-force** options.

It is possible to specify different properties such as minimum and maximum bitwidth or the way self gating interacts with tool-inserted clock gates, that are applicable to each type of comparison logic. This has to be specified by using separate command **set_self_gating_options**.

Please note that the option defines the type of logic to be used, but not the actual type of cell to be inserted, so for instance when **nand** is chosen the tool synthesizes the necessary cells to form a NAND comparison function, but it can be implemented using any available cells, not necessarily a NAND gate.

When either **xor**, **or** or **nand** are selected, the self-gating banks are created in such a way all registers sharing a given self gate use the same type of comparison logic. On the other hand, when **auto** is selected, the tool is allowed to mix different types of comparison logic in the same self-gating bank.

Use the **-clear** option to remove the previously stored criteria from the specified objects.

Use the **-reset** option to remove the previously stored criteria from all objects.

When the command is executed for an object that already has a previously set criterion, this is overwritten by the new setting.

For modules and power domains, the settings are stored on the hierarchical instances associated to the specified objects at the moment the command is executed. As a consequence, if for example there is a module 'bot' with instances 'b1' and 'b2', after following command sequence all objects under both hierarchical instances 'b1' and 'b2' are excluded from self gating because the second command overwrites on 'b1' the previous setting:

```
prompt> set_self_gating_objects -include {b1}
prompt> set_self_gating_objects -exclude {bot}
```

If there is no setting directly stored by the command on a particular register, the setting stored at the closest upper-level hierarchical instance, or the top level design, is applicable to it. If there is no setting stored at any upper-level object, the tool assumes that the register is included as candidate for self gating, and XOR logic is used as comparison function.

The **set_self_gating_objects** command must be run before executing the **compile** command. The options are persistent in the design library.

EXAMPLES

The following example excludes the register bank A_reg from self gating:

```
prompt> set_self_gating_objects -exclude [get_cells A_reg[*]]
```

The following example disables self gating at the top level design, but allows it for registers under hierarchical instance mid1, using OR logic as comparison function for them:

```
prompt> set_self_gating_objects -exclude [get_designs top]
prompt> set_self_gating_objects -include [get_cells mid1] -tree_type or
```

The following example removes the directive to include, force or exclude self gating from the registers in ADDER:

```
prompt> set_self_gating_objects \
      -clear [get_cells {ADDER/out1_reg[*] ADDER/out2_reg[*]}]
```

The following example removes the directive to include, force or exclude self gating from all objects:

```
prompt> set_self_gating_objects -reset
```

The following example forces self gating on module mid, chooses NAND logic as the comparison function and uses **set_self_gating_options** command to specify the bank sizes that are applicable in that case:

```
prompt> set_self_gating_objects -force mid -tree_type nand
prompt> set_self_gating_options -objects mid \
                  -min_bitwidth 3 \
                  -max_bitwidth 6 \
                  -tree_type nand
```

The following example lets the tool decide automatically what type of combinational logic is used as comparison function for all registers, and uses **set_self_gating_options** command to specify the interaction with clock gating and the bank sizes that are applicable in that case:

```
prompt> set_self_gating_objects -tree_type auto -include [all_registers]
prompt> set_self_gating_options -interaction_with_clock_gating collapse \
  -min_bitwidth 4 -max_bitwidth 8 -tree_type auto
```

SEE ALSO

compile_fusion()
set_self_gating_options(2)

set_self_gating_options

Controls the structural options for self gating insertion.

SYNTAX

status **set_self_gating_options**
[-min_bitwidth *bitwidth*]
[-max_bitwidth *bitwidth*]
[-interaction_with_clock_gating *interaction_type*]
[-tree_type *combinational_logic*]
[-objects *object_list*]

Data Types

bitwidth integer
interaction_type none | insert | collapse
combinational_logic xor | or | nand | auto
object_list list

ARGUMENTS

-min_bitwidth *bitwidth*

Specifies the minimum accumulated bitwidth that a self gate can gate. It is an integer value, greater than or equal to one. This setting is optional, and if not set it defaults to 4. The specified value must be less than or equal to the max bitwidth value.

-max_bitwidth *bitwidth*

Specifies the maximum accumulated bitwidth that a self gate can gate. It is an integer value, greater than or equal to one. This setting is optional, and if not set it defaults to 8. The specified value must be higher than or equal to the min bitwidth value.

-interaction_with_clock_gating *interaction_type*

Specifies how to interact with the presence of tool-inserted clock gates when inserting self gating. The value should be one of the following: **none**, **insert** or **collapse**. If set to **none**, registers that are already clock gated by the tool are discarded from self gating insertion. If set to **insert**, registers that are already clock gated by the tool are valid candidates for self gating insertion. If set to **collapse**, registers that were already clock gated by the tool are valid candidates for self gating insertion, and when this is done, the tool collapses the clock gate and the self gate into one cell which adopts properties from both self gates and regular clock gates. By default, value **none** is assumed.

-tree_type *combinational_logic*

Specifies that the self-gating options are applied when the register being self gated is configured to use the specified combinational logic as comparison function. The value should be one of the following: **xor**, **or**, **nand** or **auto**. This configuration is done along with the **set_self_gating_objects** command. By default, value **xor** is assumed.

-objects ***object_list***

Specifies that the self-gating options are applied to the specified list of design objects. Valid objects are hierarchical instances, power domains and modules. If this option is not used, the self-gating options are set for the entire design.

DESCRIPTION

Each instance of this command controls the structural options for self gating insertion, such as minimum/maximum bitwidth or the way it should interact with tool-inserted clock gates, that are applicable to all objects using a given type of comparison logic.

Use the **-min_bitwidth** and **-max_bitwidth** options to specify the expected bitwidth range that is applicable to all registers configured to use the type of comparison logic specified by option **-tree_type**, that are under the hierarchical objects specified by option **-objects**.

Use the **-interaction_with_clock_gating** option to specify the way self gating should behave for registers that are already clock gated by the tool, that is applicable to all registers configured to use the type of comparison logic specified by option **-tree_type**, that are under the hierarchical objects specified by option **-objects**.

When the command is executed for an object that already has options stored for the same type of comparison logic as the one specified by the current instance of the command, the previous options are overwritten by the new ones.

For modules and power domains, the options are stored on the hierarchical instances associated to the specified objects at the moment the command is executed. As a consequence, if for example there is a module 'bot' with instances 'b1' and 'b2', after following command sequence all objects under both hierarchical instances 'b1' and 'b2' are configured to form self-gating banks with bitwidth in the range 4-6, because the second command overwrites on 'b1' the previous setting:

```
prompt> set_self_gating_options -min_bitwidth 3 \
                    -max_bitwidth 5 \
                    -objects {b1}
prompt> set_self_gating_options -min_bitwidth 4 \
                    -max_bitwidth 6 \
                    -objects {bot}
```

If there is no setting directly stored by the command on a particular hierarchical instance for a given type of comparison logic, the setting stored at the closest upper-level hierarchical instance, or the top level design, is applicable to it. If there is no setting stored for that type of comparison logic at any upper-level object, the tool assumes a minimum bitwidth of 4, a maximum bitwidth of 8 and interaction with clock gating **none** for that type of comparison logic.

The options are persistent in the design library.

EXAMPLES

The following example specifies the self-gating options described below:

```
prompt> set_self_gating_options -min_bitwidth 6 \
                    -max_bitwidth 8
The following self gating options for xor-based tree type are applied to: top-level (design)
Minimum bitwidth: 6
Maximum bitwidth: 8
Interaction with Clock Gating: none
1
```

```
prompt> set_self_gating_options -min_bitwidth 3 \
                    -max_bitwidth 5 \
                    -interaction_with_clock_gating insert \
                    -tree_type or
The following self gating options for or-based tree type are applied to: top-level (design)
Minimum bitwidth: 3
Maximum bitwidth: 5
Interaction with Clock Gating: insert
1

prompt> set_self_gating_options -min_bitwidth 4 \
                    -max_bitwidth 6 \
                    -interaction_with_clock_gating collapse \
                    -tree_type nand
The following self gating options for nand-based tree type are applied to: top-level (design)
Minimum bitwidth: 4
Maximum bitwidth: 6
Interaction with Clock Gating: collapse
1

prompt> set_self_gating_options -min_bitwidth 3 \
                    -max_bitwidth 8 \
                    -interaction_with_clock_gating insert \
                    -tree_type auto
The following self gating options for auto-based tree type are applied to: top-level (design)
Minimum bitwidth: 3
Maximum bitwidth: 8
Interaction with Clock Gating: insert
1
```

The following examples specifies that, when comparison logic **xor** is selected (the default), the tool must create self-gating banks in the bitwidth range 3-6 for the whole design, except for the objects under mid1, for which the specified range is 4-8.

```
prompt> set_self_gating_options -min_bitwidth 3 \
                    -max_bitwidth 6
The following self gating options for xor-based tree type are applied to: top-level (design)
Minimum bitwidth: 3
Maximum bitwidth: 6
Interaction with Clock Gating: none
1

prompt> set_self_gating_options -objects [get_cells mid1] \
                    -min_bitwidth 4 \
                    -max_bitwidth 8
The following self gating options for xor-based tree type are applied to: mid1 (hier inst)
Minimum bitwidth: 4
Maximum bitwidth: 8
Interaction with Clock Gating: none
1
```

SEE ALSO

set_self_gating_objects(2)
report_clock_gating(2)

set_sense

Specifies the unateness propagating forward for pins with respect to the clock source.

SYNTAX

string **set_sense**
[-type *type* | -stop_propagation | -clock_leaf | -positive |
-negative | -non_unate | -pulse *pulse_type*]
[-clocks *clock_list*]
object_list

Data Types

type string
pulse_type string
clock_list list
object_list list

ARGUMENTS

-type clock | data

Specifies whether the type of sense being applied refers to clock networks or data networks. The possible values for the type variable are *clock* and *data*. Note that if **-type** data is given, the **-stop_propagation** and **-clocks** options must be used as well. By default, the **-type** is *clock*.

-stop_propagation

Stops the propagation of specified clocks in the *clock_list* from the specified pins or cell timing arcs in *object_list*. Only clock used as clock is stopped if used with **-type clock**. To stop propagation for both clock as clock and clock as data, you need two commands: one with **-type clock** and one with **-type data**. The cell timing arcs are not supported in the *object_list* if **-type** data is used. **-stop_propagation** cannot be specified with **-positive**, **-negative** or **-pulse**.

-clock_leaf

Stops the propagation of specified clocks in the \fclock_list from the specified pins in the \fobject_list. Also, it specifies the pin as a clock consumer, so that the clock branches reaching this pin will be considered as clock consumer as well. In other words, the attribute "is_clock_used_as_clock" will be true for all the pins of the clock branches that reach the clock leaf pin. You cannot use the **-clock_leaf** option with the **-positive**, **-negative**, **-stop_propagation** or **-pulse** options.

-positive

Specifies positive unateness applied to all pins in *object_list* with respect to the clock source. **-positive** cannot be specified with **-negative** or **-non_unate** or **-pulse** or **-type data**.

-negative

Specifies negative unateness applied to all pins in *object_list* with respect to the clock source. **-negative** cannot be specified with **-positive** or **-non_unate** or **-pulse** or **-type data**.

-non_unate

Launches both the positive-unate sense and the negative-unate sense of the clock from the clock source. If you use the **-non_unate** option, you must also use the **-clocks** option. Please note that this option is only valid for clock source pins/ports. **-non_unate** cannot be specified with **-positive** or **-negative** or **-pulse** or **-type data**.

-pulse *pulse_type*

Specifies the type of pulse clock applied to all pins in *object_list* with respect to the clock source. The possible values for pulse_type are: rise_triggered_high_pulse, fall_triggered_high_pulse, rise_triggered_low_pulse, and fall_triggered_low_pulse. **-pulse** cannot be specified with **-negative** or **-positive** or **-type data**.

-clocks *clock_list*

Specifies a list of clock objects to be applied with the given unateness that is placed on all pin objects in *object_list*. By default, all clocks passing through the given pin objects are considered.

object_list

Specifies the list of pins or cell timing arcs with specified unateness to propagate. The timing arcs object can be used only with the **-stop_propagation** option. The cell timing arcs are not supported in the *object_list* if **-type** data is used.

DESCRIPTION

This command restricts unateness at pin (to positive or negative unate) with respect to the clock source. However, the specified unateness only applies within the non-unate clock network. In this case, user-defined sense propagates forward from the given pins.

If the **-clocks** option is specified, only the specified clock domains are applied. Otherwise, all clocks passing through the given pin objects will be considered.

The tool issues a warning message if the specified sense on given pins cannot be respected in case there is predefined unateness for given pins. Hierarchical pins are not supported.

In addition, you can completely stop propagation of certain clocks in clock networks or data networks by using **-stop_propagation** option together with the **-type** option.

Use the **remove_sense** command to undo **set_sense** settings.

Multicorner-Multimode Support

This command works only on the current mode.

EXAMPLES

The following example specifies a positive unateness for a pin named **XOR/Z** with respect to clock **CLK1**.

```
prompt> set_sense -positive -clocks [get_clocks CLK1] XOR/Z
```

The following example specifies negative unateness for a pin named **MUX/Z** for all clocks.

prompt> **set_sense -negative MUX/Z**

The following example stops clock propagation in data networks from **AND/Z** for clock **CLK1**.

prompt> **set_sense -type data -stop_propagation -clocks CLK1 AND/Z**

SEE ALSO

remove_sense(2)

set_serialize_configuration

Specifies the serializer configuration for the design.

SYNTAX

status **set_serialize_configuration**
[-inputs *number_of_deserializer_inputs*]
[-outputs *number_of_serializer_outputs*]
[-update_stage true | false]
[-exclude_clocks *list_of_clock_names_to_exclude*]
[-serializer_clock *clock_for_serializer_controller*]
[-wide_duty_cycle true | false]

Data Types

inputs integer
outputs integer
parallel_mode string
test_mode string

ARGUMENTS

-inputs *number_of_deserializer_inputs*

Specifies the number of inputs used to load scan data in the deserializer for scan compression. The inputs are a subset of the scan inputs used for base mode (reconfigurable scan mode).

The default is equal to the minimum number of chains that DFT would build if scan compression were not enabled.

-outputs *number_of_serializer_outputs*

Specifies the number of outputs used to observe scan data from the serializer for scan compression. The outputs are a subset of the scan outputs used for base mode (reconfigurable scan mode).

The default is equal to the minimum number of chains that DFT would build if scan compression were not enabled.

-update_stage true | false

Adds an update stage to the deserializer.

The default is **false**.

-wide_duty_cycle true | false

When set to **true**, the serializer clock controller generates the internally generated scan shift clocks and the update stage clock with a duty cycle closer to 50%, which helps prevent timing issues.

The default is **false**, which generates these clocks by passing through selected high-frequency clock pulses from the clock source.

-exclude_clocks ***list_of_clock_names_to_exclude***

Specifies the list of clock names to be excluded from gating in the serializer clock controller. The specified clock must be previously declared as a test clock using the **set_dft_signal** command.

The default is null (no value specified).

-serializer_clock ***clock_for_serializer_controller***

In hierarchical flows, when **set_scan_compression_configuration -serialize** is set to **core_level**, you can specify the clock to be used for the deserializer-serializer. The specified clock must be previously declared as a test clock using the **set_dft_signal** command.

The default is null (no value specified).

DESCRIPTION

The **set_serialize_configuration** command allows you to specify certain DFTMAX insertion parameters prior to using the **insert_dft** command.

The command also enables the configuration of the number of serial scan inputs and serial scan outputs that are to be used for a particular serial scan compression mode.

The other parameters are used to specify the clocks that can be excluded from clock-gating using the **-exclude_clocks** option, as well as the serializer clock in core-level flows using the **-serializer_clock** option.

An update stage can be inserted along with the deserializer to enable fast clocking schemes when the **-update_stage** option is set to **true**.

SEE ALSO

define_test_mode(2)
insert_dft(2)
preview_dft(2)
report_dft(2)

set_shaping_group_order

Adjusts the order of element within a shaping groups.

SYNTAX

none **set_shaping_group_order**
[-object *group_object*]
[-before *group_object*]
[-after *group_object*]
[-move_left]
[-move_right]
[-make_first]
[-make_last]
[-set *order*]
shaping_group

Data Types

group_object object
order list of objects
shaping_group object

ARGUMENTS

-object *group_object*

Specifies the object within the group to reorder.

-before *group_object*

Specifies an object within the group to place the target object immediately to the left of.

-after *group_object*

Specifies an object within the group to place the target object immediately to the right of.

-move_left

Moves the target object one space to the left. If the target object is already at the front of the group, no change is performed.

-move_right

Moves the target object one space to the right. If the target object is already at the end of the group, no change is performed.

-make_first

Moves the target object to the front of the group.

-make_last

Moves the target to the end of the group.

-set *order*

Sets the order of the group to the given value. The list of objects must consist of exactly the objects that exist in the group.

shaping_group

Specifies the shaping group on which to perform the ordering operation.

DESCRIPTION

The command modifies the order of elements with a shaping group. You may move an individual object using the basic move arguments or provide a new order for the entire group. The modifications are undoable.

EXAMPLES

The following commands show the effect of running the **set_shaping_group_order** command on a sample shaping group.

prompt> **set SG [create_group -shaping -name mySG "U1 U2 U3 U4 U5"]**
prompt> **get_attribute -name objects $SG**
{U1 U2 U3 U4 U5}
prompt> **set_shaping_group_order -object U1 -make_last $SG**
prompt> **get_attribute -name objects $SG**
{U2 U3 U4 U5 U1}
prompt> **set_shaping_group_order -object U4 -move_left $SG**
prompt> **get_attribute -name objects $SG**
{U2 U4 U3 U5 U1}
prompt> **set_shaping_group_order -set "U4 U2 U1 U3 U5"**
prompt> **get_attribute -name objects $SG**
{U4 U2 U1 U3 U5}

SEE ALSO

create_group(2)
group(3)

set_shaping_options

Specifies the shaping and placement options to apply with the **shape_blocks** command. Options specified using the **set_shaping_options** command are persistent throughout the session and are changed only by a subsequent **set_shaping_options** command. Options specified directly to the **shape_blocks** command are not persistent.

SYNTAX

int **set_shaping_options**
[-min_channel_size *size*]
[-guard_band_size *guard_band*]
[-utilization_slack *slack_value*]
[-keep_top_level_together true | false]
[-add_channel_blockages none | partial | soft | hard]
[-reset]

Data Types

size float
guard_band float
slack_value float

ARGUMENTS

-min_channel_size *size*

Specifies the minimal required channel size, in microns, between the following objects:

- Hierarchical cell (block) to hierarchical cell
- Hierarchical cell and fixed top-level macro
- Hierarchical cell and core boundary

This option is valid only when you specify **shape_blocks -channels**. By default, the minimal channel size is 0.

-guard_band_size *guard_band*

Specifies the guard band size, both width and height, for the voltage areas created by the **shape_blocks** command. The units are microns. When the **shape_blocks** command encounters a power domain with no voltage area, it shapes the power domain and creates the corresponding voltage area with the specified guard band size. The **shape_blocks** command will not change the guard band size of existing voltage areas. By default, the guard band size is 0.

-utilization_slack *slack_vale*

Specifies the utilization slack to use during incremental shaping.

During incremental shaping, utilization slack specifies the allowable difference between actual block utilization and target utilization. For example, consider a design with a target utilization of 0.7. The design contains two blocks, BLOCKA and BLOCKB, with utilizations of 0.8 and 0.7 respectively. Without utilization slack, the utilization for BLOCKA cannot be reduced because area cannot be "borrowed" from any other block; there are no other blocks with utilization less than 0.7.

With utilization slack, the utilization for BLOCKB can be increased above the target utilization in order to reduce the utilization for BLOCKA. For example, the tool can increase the utilization for BLOCKB from 0.7 to 0.75, then reduce the utilization for BLOCKA from 0.8 to 0.75 (if both blocks have the same size). If BLOCKB is significantly larger than BLOCKA, utilization slack allows the tool to add a small increase in utilization for BLOCKB to greatly reduce the utilization for BLOCKA.

By default, the utilization slack value is 0.1 (10 percent).

-keep_top_level_together true | **false**

Specifies whether the placement regions of the top level cells are kept together during non-incremental shaping. By default, the setting is true such that top level cells are not separated by blocks.

-add_channel_blockages none | **partial** | **soft** | **hard**

Specifies the type of placement blockages to insert into congested channels by using the **shape_blocks -incremental congestion_driven** command. By default, the command inserts soft blockages.

-reset

Resets all shaping options to their defaults.

DESCRIPTION

This command specifies the shaping options for the current session. The shaping options are applied by the **shape_blocks** command.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following example sets the minimal required channel size to 50 microns.

prompt> **set_shaping_options -min_channel_size 50**

The following example sets the guard band size to 5 microns for the voltage areas created by the **shape_blocks** command.

prompt> **set_shaping_options -guard_band_size 5**

SEE ALSO

shape_blocks(2)

set_signal_io_constraints

Sets constraints for the placement of I/O driver pads within an I/O guide.

SYNTAX

status **set_signal_io_constraints**
[-io_guide_object *io_guide*]
[-constraint *io_guide_constraints*]
[-file *file_name*]
[-csv *csv_file_name*]
[-map_file *map_file_name*]

Data Types

io_guide list
io_guide_constraints string
file_name string
csv_file_name string
map_file_name string

ARGUMENTS

-io_guide_object *io_guide*

Specifies the name of a single I/O guide or a collection that contains a single I/O guide. You can use the **get_io_guides** command to return the I/O guide object, however, the collection must contain only one I/O guide.

-constraint *io_guide_constraints*

Specifies the I/O constraint. This option must be used with **-io_guide_object** option. It describes the detailed constraints for the guide specified by **-io_guide_object** option. There are three types of signal constraints: order-based constraints, location constraints, and boundary-spacing control constraints.

The first type of signal I/O constraint is an order-based constraint. An order-based constraint specifies the relative order of I/O driver pads within the I/O guide. This constraint is described by an ordered sequence of basic elements, which consist of I/O driver pads and constraint keywords or values. There are four basic formats that can be used to describe each element. The first order-based constraint format uses the order_only keyword in curly brackets followed by a sequence of pad cell names. The {order_only} keyword and pad cell names are surrounded by curly brackets. The pads must be placed in the specified order, although additional spacing and other pad cells can be inserted between adjacent pads. An example format is as follows:

{{order_only} pad1 pad2 pad3}

The second order-based constraint format contains a floating value within curly brackets, which represents white space without any pads placed. An example format is as follows:

{*spacing_value*}

The third order-based constraint format contains a sequence of pad names and spacing values to indicate the pad ordering and spacing. If no spacing value is specified between two pads, the two pads are placed with no gap between them. An example format is as follows:

{pad1 spacing1 pad2 spacing2 pad3 spacing3 pad4}

The fourth order-based constraint format contains a pitch value in curly brackets followed by a sequence of pad instance names. The pads are placed in the given order with the specified pitch. An example format is as follows:

{{pitch} pad1 pad2 pad3}

The order-based constraint can contain several elements in any of the four formats. All elements are placed in outer curly brackets. Additional space and pads can be inserted between adjacent elements. If users do not allow anything between adjacent elements, they should merge those elements into a single element. Alternatively, they can put those elements into a pair of curly brackets. In the following example, pad3 and pad4 will be placed next to each other.

{ {{50} pad1 pad2 pad3} {pad4 20 pad5} }

The order-based constraint follows a clockwise orientation. The pad instances at the front of the list are placed closer to the starting point of the I/O guide than later pad instances.

The second type of signal I/O constraint is a location constraints. The format begins with the keyword {absolute} in curly brackets, followed by a pad name and an offset value. The offset is with respect to the starting point the I/O guide. The constraints of fixed locations are independent of constraints in the other four formats. An example format is as follows:

{{absolute} pad offset}

You can specify multiple fixed pad locations as follows:

{{absolute} pad1 offset1}{{absolute} pad2 offset2}

The third type of signal I/O constraint is boundary-spacing control constraints. The format begins with the {no_space} keyword in curly brackets, followed by a **begin**, **end**, or **both** keyword. Use this constraint type to determine whether space can be added between the starting-point of I/O guide and the first I/O pad, or between the ending-point of I/O guide and the last pad. If this constraint is not used, space can be added. Note that if a corner cell is already placed in the I/O guide, the pad will not overlap with the corner cell. An example format is as follows:

{{no_space} both}

The boundary-spacing control constraint and order-based constraint can be combined to achieve the same effect of hard location constraints. The pad locations are described using the gap between the I/O guide starting point and the first pad and the gaps between all adjacent pads. Specifically, you must create one and only one element in the third order-based constraint format mentioned above. In addition, they add the following:

{{no_space} begin}

The location constraint and order-based constraint can be combined. For instance, to place a sequence of three pads (pad1, pad2, and pad3) with a gap of 50 microns between adjacent pads and starting at a guide offset of 700 microns, the following two constraints can be used. The location constraint specifies the first pad location. The order-based constraint describes the sequence and relative locations among all pads.

{pad1 50 pad2 50 pad3}

{{absolute} pad1 700}

It is important to point out that the spacing in order-based constraint specifies an empty gap in which no pad can be placed, including any fixed pads not in any constraints. No power pads can be inserted into the gap either. To create a gap in which pads can be inserted later, you must use a location constraint. The following constraints create a gap of 500 micron in which other pads can be placed, assuming that the width of pad1 is 5 micron and there is no other constraint.

{{absolute} pad1 500}{{absolute} pad2 1005}

The following constraints create a gap of 500 micron in which no pads can be inserted. The locations of the two pads are the same as the example above.

{{absolute} pad1 500}

{pad1 500 pad2}

Only a single **-constraint** option is used for each I/O guide. As a result, the last **set_signal_io_constraints** command overrides all previous commands if they are applied to the same I/O guide. The specification following the **-constraint** option must be placed in a pair of curly brackets.

-file ***file_name***

Specifies the name of the file that contains I/O signal constraints for one or more I/O guides. For each I/O guide, the specification begins with the guide name, followed by the I/O signal constraint. The format is the same as that in the **-constraint** option. A semicolon (;) terminates the I/O signal constraint for the I/O guide.

You can add comments in a signal I/O constraint file. Each comment line must start with a # character or // characters. The entire line must be either comments or constraints. Currently, the combination of comments and constraints in a single line is not supported.

-csv ***csv_file_name***

Specifies the name of the comma-separated values file that contains I/O signal constraints. You must specify the **-csv** option together with the **-io_guide_object** option to assign the I/O guide to use. Only one option, **-constraint** or **-csv**, can be used along with the specified I/O guide.

The CSV file specified by this option should contain the constraint data in the form of comma-separated values. The two required field names are "Pad Name" and "Offset Value". The first line in the CSV file must contain all the field names and the field names can appear in any order. If a required field is missing, the command issues an error message and exits. The CSV file can contain additional fields, however, unsupported field names are ignored and no error message is issued. All entries in a specific line must be separated by a comma. You can specify the following kinds of constraints using the CSV file:

1) Order only constraints: In this type of constraints, the specified pads must be placed in the order in the file, although additional pad cells and other pads might be inserted between adjacent pad cells. For order only constraints the "Offset Value" field should not be specified.

2) Location constraints: Constraints are specified as pairs of pad name and offset values. The offset values are based on the starting point of the I/O guide. This ensures that during I/O placement the pad will be placed at its desired offset on the I/O guide.

Note the following restrictions for the contents of the CSV file:

- No standard field should be repeated.
- The number of fields should be equal for each line in the file.
- There should be no missing or extra field entries in each line. For example, if you specify an offset value for one of the pads in the CSV file, an offset value must be specified for all other pad names in the file.

-map_file ***map_file_name***

Specifies an optional CSV file which maps custom column names in the connections file to standard Synopsys column names. The format for each line is <custom_name>,<standard_name>.

For example, the map file contents could be as follows:

Cell Name,Pad Name
Distance,Offset Value

Where "Cell Name" and "Distance" are custom column names which can be used in the constraint CSV file. The custom column names should not be same as the standard column names.

DESCRIPTION

This command specifies how pad instances are placed within their I/O guides.

If a pad instance described in the I/O signal constraints is not assigned to any I/O guides, it will be assigned to the I/O guide. Otherwise, the command will error out if assigned to another I/O guide. If a pad instance assigned to the I/O guide is not part of the I/O signal constraint, it will be placed without violating the constraint.

EXAMPLES

The following example forces a specific order of placement of four pad instances in the I/O guide named "north". The actual spacing between adjacent pads is determined by the I/O placer.

```
prompt> set_signal_io_constraints -io_guide_object north \
  -constraint {{pad1} {pad2} {pad3} {pad4}}
```

The following example forces a specific order of placement of four pad instances in I/O guide named "north". The first two pads are placed next to each other. There is a 100 micron spacing between the last two pads. The spacing between the second pad and third pad is determined by the I/O placer.

```
prompt> set_signal_io_constraints -io_guide_object north \
  -constraint {{pad1 pad2} {pad3 100 pad4}}
```

The following example forces a specific order of placement of four pad instances in I/O guide named "north". The pitch of all pads is 200 micron.

```
prompt> set_signal_io_constraints -io_guide_object north \
  -constraint {{200} pad1 pad2 pad3 pad4}
```

The following example forces a specific placement of three pad instances in I/O guide named "north". All pad locations are exactly calculated based on the spacing in the constraints. Specifically, the distance before pad1 is 20 micron. The gap between pad1 and pad2 is 45 micron. Pad2 and pad3 are abutted. There might be space after pad3 if the I/O guide is very wide.

```
prompt> set_signal_io_constraints -io_guide_object north \
  -constraint {{{no_space} begin} {20 pad1 45 pad2 pad3} }
```

SEE ALSO

place_io(2)
remove_signal_io_constraints(2)
report_signal_io_constraints(2)

set_signoff_check_drc_icv_live

Sets ICV-Live rules to check.

SYNTAX

set_signoff_check_drc_icv_live
[-reset]
[-select_rules *rule_names*]
[-unselect_rules *rule_names*]
[-description *description*]

Data Types

rule_names	list
description	string

ARGUMENTS

-reset

Reset ICV-Live manager and check all rules with ICV-Live run. If specified, a reset is the first action to be performed.

-select_rules *rule_names*

Specifies the list of rule names to check with ICV-Live run. Mutually exclusive with the -unselect_rules argument.

-unselect_rules *rule_names*

Check all rules with ICV-Live run except this specified list of rule names. Mutually exclusive with the -select_rules argument.

-description *recipe/preset_description*

Specifies the description of set of rules to check with ICV-Live

DESCRIPTION

This command specifies the rules to check with ICV-Live run.

By default, ICV-Live checks all run-set rules.

You can use this command to setup the rules to check.

EXAMPLES

The following example setup specific rules to check.

prompt> **set_signoff_check_drc_icv_live -select_rules { "M2.DN.1" "DM1.R.1" "M1.A.1" "M1.A.2" }**

SEE ALSO

signoff_check_drc_icv_live(2)
signoff.check_drc_live.runset(3)
signoff.check_drc_live.run_dir(3)
signoff.check_drc_live.default_layers(3)

set_simstate_behavior

Specify the simulation simstate behavior for a model or library.

SYNTAX

status **set_simstate_behavior**
<ENABLE | SIMSTATE_ONLY | PORT_CORR_ONLY | DISABLE>
[-lib *name*]
[-models *model_list*]
[-elements *element_list*]
[-exclude_elements *exclude_list*]

Data Types

name	string
model_list	list
element_list	list
exclude_list	list

ARGUMENTS

<ENABLE | SIMSTATE_ONLY | PORT_CORR_ONLY | DISABLE>

Define the type of simulation behavior for the specified model(s).

-lib *name*

The library name.

-model *model_name*

One or more model names.

-elements *element_list*

A list of instances.

-exclude_elements *exclude_list*

A list of instances to exclude from the effective_element_list.

DESCRIPTION

This command specifies the simulation behavior for models or instances.

This command is for functional verification only does not have any impact on the implementation tools.

Implementation tool will not parse and check the options and it will not preserve this command in save_upf.

EXAMPLES

```
prompt> set_simstate_behavior DISABLE \
        -lib library1
        -models ANDX7_power_aware
```

SEE ALSO

sim_corruption_control(2)

set_site_array_stack_order

Sets the stack order of a site array in the current design.

SYNTAX

collection **set_site_array_stack_order**
[-above *site_array*]
[-below *site_array*]
[-raise]
[-lower]
[-top]
[-bottom]
site_array

Data Types

site_array string or collection

ARGUMENTS

-above *site_array*

Specifies the reference site array above which to place the given site array. This option is mutually exclusive with **-below**, **-raise**, **-lower**, **-top** and **-bottom**.

-below *site_array*

Specifies the reference site array below which to place the given site array. This option is mutually exclusive with **-above**, **-raise**, **-lower**, **-top** and **-bottom**.

-raise

Specifies that the given site array is to be raised by one position. This option is mutually exclusive with **-above**, **-below**, **-lower**, **-top** and **-bottom**.

-lower

Specifies that the given site array is to be lowered by one position. This option is mutually exclusive with **-above**, **-below**, **-raise**, **-top** and **-bottom**.

-top

Specifies that the given site array is to be placed at the top. This option is mutually exclusive with **-above**, **-below**, **-raise**, **-lower** and **-bottom**.

-bottom

Specifies that the given site array is to be placed at the bottom. This option is mutually exclusive with *-above*, *-below*, *-raise*, *-lower* and *-top*.

site_array

Specifies the site array, either a name or collection, whose stack order needs to be modified.

DESCRIPTION

The **set_site_array_stack_order** command modifies the stacking order of a site array. The given site array can be moved to the top or bottom of the stack, can be raised or lowered by one position from its current stack position or can be moved above or below a specified reference site array. All of these options are mutually exclusive.

This re-ordering of the stack can cause effective region and constituent site row recomputation of all site arrays if the given site array is opaque. The given site array's site rows will also be recalculated.

EXAMPLES

The following example moves a site array named 'SA1' to the top of the stack.

```
prompt> set_site_array_stack_order -top SA1
```

The following example moves the site array named 'SA1' below another site array named 'SA2'.

```
prompt> set_site_array_stack_order -below SA2 SA1
```

SEE ALSO

create_site_array(2)
get_site_arrays(2)
remove_site_arrays(2)

set_size_only

Controls whether the specified leaf cells can only be sized during optimization.

SYNTAX

status **set_size_only**
[-all_instances]
cell_list
[true | false]

Data Types

cell_list collection

ARGUMENTS

-all_instances

Controls how the size-only setting is applied to child instances of multiply instantiated designs.

When you use this option, if a specified cell is in a multiply instantiated instance of a subdesign, the tool applies the size- only setting to the similar cells in all instances of that sub- design.

This is especially useful to avoid changing the current design to issue this command in different designs.

cell_list

Specifies the leaf cells on which to apply the size-only setting.

true | **false**

Specifies the value of the size-only setting.

The default is **true**, so **set_size_only U1** has the same effect as **set_size_only U1 true**.

DESCRIPTION

This command applies size-only settings to the specified leaf cells.

When you apply a size-only setting of **true** to leaf cell, optimization can perform only sizing operations on that cell. A setting of **false** removes the size-only settings for the specified leaf cells, unless the presence of constraints on the cell is also causing it to be size_only.

For example, if apply a size-only setting of **true** to a cell, the tool attempts to optimize this cell by replacing it with a cell of the same function. The size-only setting on a cell does not prevent changes to the net driven by this cell. For example buffers can be added to this net. To prevent changes to the net, use the **set_dont_touch** command to set the **dont_touch** attribute on the net.

Please note, size_only can not be set on unmapped DesignWare cell. Before compile, unmapped DesignWare cells appear to be leaf cells. However, after compile, they will become hierarchical cells. Currently we do not allow setting size_only on unmapped DesignWare cells.

To report the size-only settings, use the **report_size_only** or **report_cell** commands.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example applies a size-only setting of **true** to the cell named U1.

```
prompt> set_size_only U1
```

The following example removes the size-only setting from the cell named U1.

```
prompt> set_size_only U1 false
```

The following example uses the **-all_instances** option.

```
prompt> set_size_only -all_instances mid1/bot1/c1
Information: '3' cells set to 'size_only'. (CSTR-020)
1
```

SEE ALSO

report_size_only(2)
report_cell(2)
get_attribute(2)

set_skew_macros

Specifies the macros for which slack needs to be improved via skew.

SYNTAX

int **set_skew_macros**
-bank_name *name*
-macros *macro_list*
[-improve_side *input*|*output*|*both*]

Data Types

macro_list collection

ARGUMENTS

-bank_name *name*

Specifies a unique name for the collection of related macros to be skewed. If the bank name is already used in some prior specification of set_skew_macros, the old specification is overridden.

-macros *macro_list*

Specifies a collection of macros in the bank. A macro can be in at most one bank. If a macro in the collection was in a prior **set_skew_macros** command, then that macro is removed from the prior bank and added to the new bank.

-improve_side *input* | *output* | *both*

Specifies which side of the macro needs to be improved through skew. The default is *output*.

DESCRIPTION

The **set_skew_macros** command specifies a set of macros for which slack needs to be improve by skew.

Macros such as RAMs have complex timing paths around them that need clock skewing to achieve the target period. Traditionally, this is done by using the set_clock_latency commands on macros and/or registers around the macros, to provide more time to close timing on macro paths. This method is tedious due to the requirement to understand the topology around macros and to compute the skew values.

The **set_skew_macros** command specifies a bank which is a collection of related macros (e.g. data RAMs where each RAM represents a specific bit range such as data_ram[31:0], data_ram[63:32]). It also specifies which side of the bank needs to be improved through skew.

The tool automatically detects the best way to improve the failing timing paths. For example, if improve_side is input, to the slack on the macro's inputs can be improved by preponing the input registers, postponing the macros, and/or postponing the output registers. The tool makes such decisions along with the amount of skew.

EXAMPLES

The example below specifies a bank 'data1' with macros named '*u_data_ram*' and the improve_side to be input side of the bank.

prompt> **set_skew_macros -bank_name data1 -macros [filter_coll**
[get_cells -hier -filter is_hard_macro] full_name=~*u_data_ram*] -improve_side input

SEE ALSO

remove_skew_macros(2)
report_skew_macros(2)

set_slice_preservation

Specifies which buses of registers should be optimized with slice-preservation

SYNTAX

status **set_slice_preservation**
[-bus *list_of_buses*]
[-exclude]
[-register *list_of_registers*]
[-slack_threshold *percentage_value*]
[-verbose]

Data Types

list_of_buses list of cell_bus objects
list_of_registers list of cell objects
percentage_value float

ARGUMENTS

-bus *list_of_buses*

Specifies the specific cell buses on which this command will act. If both *-bus* and *-register* are omitted, then this command will act on all well-formed register buses in the design of size greater than or equal to 16.

-exclude

Mandates that slice-preservation optimization should not be done on the indicated buses.

-register *list_of_registers*

Specifies the registers upon whose cell buses this command will act. If both *-bus* and *-register* are omitted, then this command will act on all well-formed register buses in the design of size greater than or equal to 16.

-slack_threshold *percentage_value*

If given, during timing optimization the *compile* command will disable slice preservation for any buses whose normalized negative slack is less than this threshold value. We define the normalized slack as follows: For a given path between startpoint S and endpoint P, let NS be the most negative slack at P. Let RT be the required time at P, and let AT be the arrival time at S. To normalize NS we divide it by (RT - AT): the difference between the required time at the path endpoint and the arrival time at the path startpoint. The percentage_value is thus the ratio between the negative slack and the maximum amount of time a path from S to P can take before it becomes critical.

-verbose

Mandates that the command print information about its actions.

DESCRIPTION

The *set_slice_preservation* command adds a uniformity requirement to register mapping in compile. This directive specifies that some specific buses of registers will be mapped uniformly, if possible. In this case, we say that a given bus of registers must be mapped into N identical *slices*. A *slice* may be a single single-bit register or a single multi-bit register; in either case, the bus must be mapped exactly into N of these devices. We call this uniform mapping "slice-preservation," and we refer to buses of registers that have been mapped in this way to be "slice-preserved."

We require that buses which are slice-preserved should be well-formed. See the section below for a precise definition; basically this means that the bus of registers has to be mappable in a uniform way. Optimization will seek to provide a uniform implementation, but we also allow that timing optimization may selective choose to lift this requirement for buses that contain registers which are 'too critical'. That criticality is determined as a percentage of the normalized negative slack, as described above for the *-slack_threshold* argument. During timing optimization within the *compile* command any register which is a path endpoint or startpoint for a register whose normalized slack is less than the slack threshold will no longer be slice-preserved. If the slack threshold is not given, then registers will be mapped uniformly regardless of criticality.

The *-bus* or *-register* arguments allow the caller to give bus or cell arguments. In the latter case, the bus(es) owning the indicated cell(s) will be operated on by this command. If neither argument is given, then all well-formed buses in the design will be considered for slice preservation.

The *-exclude* argument causes the given buses to not be slice-preserved. It is useful in overriding an earlier more general *set_slice_preservation*. See the EXAMPLES sections for more information on this.

Well-Formed Buses

In order for a bus of unmapped registers to be "slice-preserved" it must be implemented in a way that makes a correct uniform mapping possible. This means that all asynchronous and synchronous control pins are connected to either the same net or to the same logic value. It also means that each unique register input (typically data pins) or output (typically Q or QN pins) is connected to a bus of nets, in an increasing or decreasing ordering, without bits being unconnected, or swapped. This 'regular connection' requirement allows us to simplify the connection of identical slices.

EXAMPLES

In the following example, because neither the *-bus* nor the *-register* argument are given, slice preservation is set on all well-formed registers in the given design with size greater than or equal to 16 elements. A normalized slack threshold of -80% is given for possible register ungrouping, and the *-verbose* argument is used to get information about this buses are being slice-preserved.

```
prompt> set_slice_preservation -verbose -slack_threshold -80.0
Warning: bus abc/def_reg (width 4) will not be slice-preserved.
Information: bus gh/ij_reg (width 32) will be slice-preserved.
Information: bus qrs/tuv_reg (width 32) will be slice-preserved.
Information: bus wx/yz_reg (width 48) will be slice-preserved.
Information: Slice Preservation will be set on 3 buses
1
```

In the following example, slice-preservation is set as before, but a second invocation of the command using the *-exclude* argument disables slice preservation on a specific bus

```
prompt> set_slice_preservation -verbose -slack_threshold -80.0
Warning: bus abc/def_reg (width 4) will not be slice-preserved.
```

```
Information: bus gh/ij_reg (width 32) will be slice-preserved.
Information: bus qrs/tuv_reg (width 32) will be slice-preserved.
Information: bus wx/yz_reg (width 48) will be slice-preserved.
Information: Slice Preservation will be set on 3 buses
1

prompt> set_slice_preservation -exclude -verbose -bus gh/ij_reg
Information: bus gh/ij_reg (width 32) will not be slice-preserved.
1
```

In the following example, slice preservation is set on the bus which owns a particular register.

```
prompt> set_slice_preservation -verbose -register a/b/q_reg[0]
Information: bus q/b/q_reg (width 32) will be slice-preserved.
Information: Slice Preservation will be set on 1 bus
1
```

In the following example, slice preservation is set on the buses which are given in the list by *get_cell_buses* using the *-bus* argument

```
prompt> set_slice_preservation -verbose -bus [get_cell_buses "a*"]
Information: bus a1_reg (width 32) will be slice-preserved.
Information: bus a2_reg (width 32) will be slice-preserved.
Information: Slice Preservation will be set on 2 buses
1
```

SEE ALSO

report_cell_buses(2)

set_snap_setting

Sets snapping settings.

SYNTAX

string **set_snap_setting**
-enabled true | false
-class *class_type*
| -cursor_edge true | false
| -object_edge true | false
| -edge_radius *radius*
| -preferred_track true | false
| -fix_orientation true | false
| -macro_by_color true | false
[-snap *snap_type*]
[-user_grid *grid_name*]
[-default]

Data Types

class_type string
radius integer
snap_type string
grid_name string

ARGUMENTS

-enabled true | false

Specifies that snapping is enabled or disabled.

-class *class_type*

Specifies the object type for which to change the snapping setting. Valid class_types and object types are mapped in the following way:

- edit_group : Corresponds to edit groups
- io_cell : Corresponds to flip chip drivers, IO cells
- macro_cell : Corresponds to soft macros, hard macros, blackboxes, plan groups(partitions)
- metal_shape : Corresponds to wires, pins, shapes, vias, via matrices
- other : Corresponds to io guides, via regions

- placement_constraint : Corresponds to move bounds, shaping blockages, placement blockages, voltage areas, voltage area shapes, site arrays, site rows, rp groups, rp blockages, keepouts
- std_cell : Corresponds to standard cells
- wiring_constraint : Corresponds to routing blockages, route guides, pin guides, pin blockages, pg regions, routing corridors, routing corridor regions

-cursor_edge true | false

Specifies that the cursor should snap to visible object edges.

-object_edge true | false

Specifies that the moving or stretched object should snap to other object edges.

-edge_radius *radius*

Specifies an integer value for the search radius for snapping to object edges.

-preferred_track true | false

Specifies that only preferred tracks should be used if snapping to tracks is on.

-fix_orientation true | false

Specifies that cell orientation should be automatically fixed to match site rows or I/O guides

-macro_by_color true | false

Specifies that soft and hard macros should snap so that their colored pins are snapped to matching colored tracks.

-snap *snap_type*

Specifies the snapping type for the specified object type. Valid snap_types are:

- block
- finfet
- halfwiretrack
- litho
- mid_row_site
- row_site
- user
- wiretrack

-user_grid *grid_name*

Specifies the name of the user grid to use for snapping.

-default

Resets all options to their defaults.

DESCRIPTION

This command sets different snap settings in editing. See the argument list for details on each snapping option.

EXAMPLES

The following example enables snapping.

```
prompt> set_snap_setting -enabled true
1
```

SEE ALSO

create_grid(2)
get_snap_setting(2)

set_stage

Applies stage based settings for the current design.

This command works per stage to apply features that are flow dependent and is intended to be used with the RM flow.

SYNTAX

status **set_stage**
  [-step *step_name*]
  [-high_effort_timing]
  [-report]

Data Types

step_name string

ARGUMENTS

-step *step_name*

Specifies the step name. Valid step names are "synthesis", "placement", "cts", "post_cts_opto", "route", or "post_route". You can specify only one step name.

-high_effort_timing

Enables high-effort timing flow for the synthesis step. Only works with the "synthesis" step.

-report

Writes out the application options to the console in tabular format.

DESCRIPTION

This command applies stage-specific settings to the current design. This command is intended for the RM flow and should be used after running **set_qor_strategy** command.

The command also can read the RM Tcl variables to enable special features. If these variables are defined and set to the correct value then it enables application options for that feature. These would normally be set in the RM setup scripts to configure the RM flow. The following variables are checked:

ENABLE_PERFORMANCE_VIA_LADDER = false | true : For stage synthesis or placement, if this is true it sets the application

options **opt.common.enable_via_ladder_insertion**, **route.auto_via_ladder.update_during_route**, **place.legalize.enable_via_ladder_checks**, and **vl.flow.enable_convergence_flow** to true. It also runs the **setup_performance_via_ladder -effort high** and source a file output from the command. For non-RM users the feature can also be enabled with the Tcl variable_synopsys_rm_enable_performance_via_ladder set to true.

ENABLE_SPG = false | true : For stage placement, if this is set to true **place_opt.flow.do_spg** is set to true for the **place_opt** flow.

ENABLE_DPS = false | true : For stage placement, if this is set to true **opt.common.power_integrity** is set to true and **ccd.dps.flow_reporting** is set to {*}. This feature also requires a Digital-AF license and checks for its availability before enabling the application options. For non-RM users this feature is also enabled with the Tcl variable _synopsys_rm_enable_dps set to true.

CTS_STYLE = standard | MSCTS : For stage placement, if this is set to MSCTS then it sets the application option **place_opt.flow.enable_multisource_clock_trees** to true. It also checks if the variable TCL_REGULAR_MSCTS_FILE is defined and is pointing to a valid file path. This is sourced during place_opt.tcl.

ENABLE_HIGH_UTILIZATION_FLOW = false | true : For stage placement, in place_opt.tcl a special feature flow is enabled and this turns the **time.delay_calc_waveform_analysis_mode** to disabled.

ENABLE_IRAP = false | true : For stage post_cts_opto, if this is set to true then **opt.common.power_integrity** is set to true and **clock_opt.flow.skip_placement** is set to false to ensure placement is not skipped. This feature also requires a Digital-AF license and checks for its availability before enabling the application options. This feature also checks if the TCL_IRAP_CONFIG_FILE variable is defined and is pointing to a valid file path.

ENABLE_GRE = false | true : For stage route, if this is set to true then the **route_opt.flow.enable_power** is set to true and **time.si_enable_analysis** is set to false. The **time.si_enable_analysis** is turned to true after **clock_opt -from global_route_opt**.

ENABLE_ROUTE_AUTO_RESHIELD = after_route_opt | incremental | none : For stage post_route, if this is set to incremental then the application option **route.common.reshield_modified_nets** is set to true. It also checks if the variable ROUTE_AUTO_CREATE_SHIELDS is defined and set to either before_route_auto or after_route_auto.

ENABLE_ECO_OPT = false | true : For stage post_route, if this is set to true then it checks that the Digital-AF license is available and that the ROUTE_OPT_ECO_OPT_PT_PATH is defined and set to a valid path. No application options or commands are run but this checking is performed to avoid errors in running eco_opt later in the RM scripts.

EXAMPLES

The following example shows the usage of command in the flow.

```
prompt> set_qor_strategy -metric timing -stage pnr
prompt> set_stage -step placement
...
```

The following example applies the high effort timing flow in addition to the **compile_fusion** stage specific settings intended for the RM flow.

```
prompt> set_stage -step synthesis -high_effort_timing
```

The following example shows the RM Tcl variable ENABLE_SPG being set to true and applying the **set_stage -step placement** to enable the application option to run the Synopsys Physical Guidance flow in **place_opt**.

```
prompt> set ENABLE_SPG true
prompt> set_stage -step placement
Stage Option Version : 2.0
Step placement running
Stage-Info: Apply placement settings...
-------------------------------------
```

RM Feature: Synopsys Physical Guidance Flow = true
Report only: set_app_options -name place_opt.flow.do_spg -value true
...

SEE ALSO

set_qor_strategy(2)

set_starrc_in_design

Loads and validates a configuration file for running In-Design or fusion extraction with the StarRC engine.

SYNTAX

status **set_starrc_in_design**
-config *config_file*

-validate *true*

-mode *icc2_centric* | *starrc_centric*

Data Types

config_file string

validate boolean

mode string

ARGUMENTS

-config *config_file*

Specifies the name of the configuration file. The configuration file contains settings needed to run the StarRC tool. The syntax is described in the DESCRIPTION section.

-validate *true*

Applies the validation procedure to check if the RC network from StarRC is valid. If the RC data is invalid, StarRC In-Design fails and does not annotate the StarRC RC data onto the design. It's not applied to fusion extraction

-mode *icc2_centric* | *starrc_centric*

icc2_centric is to derive extraction options from ICC2 settings. starrc_centric is to use starrc settings from starrc command file or the starrc default settings. By default, it is icc2_centric mode. It's not applied to fusion extraction

DESCRIPTION

This command loads and validates a configuration file for running In-Design or fusion extraction with the StarRC engine. The configuration file recognizes asterisk (*) as a comment character and supports the following keywords:

- SIGNOFF_IMAGE : The location of the StarXtract executable

- CORNER_GRD_FILE : The file that lists each corner and the corresponding StarRC nxtgrd file
- MAPPING_FILE : The StarRC layer mapping file to map database physical connection layers to the layer in nxtgrd file. The same mapping file used in standalone StarRC.
- COMMAND_FILE : The StarRC command file

You can specify either MAPPING_FILE or COMMAND_FILE, however, MAPPING_FILE is the recommended setting. COMMAND_FILE is not recommended for fusion extraction as design info and setting are from ICC2/FC directly.

If a setting in the StarRC command file conflicts with the tool's setting, the tool uses its setting. For example, if the coupling capacitance threshold specified by the **set_extraction_options** command is 1.0 ff and the StarRC command file sets COUPLING_ABS_THRESHOLD to 0.5 ff, StarRC In-Design extraction uses 1.0 ff.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example specifies a configuration file that uses the MAPPING_FILE keyword to define a layer mapping file for the StarRC command.

```
prompt> set_starrc_in_design -config file1.cfg
prompt> sh cat file1.cfg

* file1.cfg
SIGNOFF_IMAGE:   /global/apps/bin/StarXtract
MAPPING_FILE:    ./tcad/nxtgrd.mapping
CORNER_GRD_FILE: ./tcad/cornerGrd.mapping
```

The following example specifies a configuration file that uses the COMMAND_FILE keyword to reference a StarRC command run file. The file contains corner and mapping file settings.

```
prompt> set_starrc_in_design -config file2.cfg
prompt> sh cat file2.cfg

* file2.cfg
SIGNOFF_IMAGE:   /global/apps/bin/StarXtract
COMMAND_FILE:    starRC.cmd
CORNER_GRD_FILE: cornerGrd.mapping
```

The following example of CORNER_GRD_FILE which defines ICC2 corner mapped with STARRC nxtgrd file

```
cold.corner /remote/TLU_GRD/cln28hp_1p10m+alrdl_7x2z_cworst.nxtgrd
hot.BC.corner /remote/TLU_GRD/cln28hp_1p10m+alrdl_7x2z_cbest.nxtgrd
hot.WC.corner /remote/pv/TLU_GRD/cln28hp_1p10m+alrdl_7x2z_cworst.nxtgrd
```

The following example of MAPPING_FILE which defines physical connnection layers in layout database to the layer in the nexgrd file. The same mapping file used in standalone StarRC.

```
conducting_layers
PO  n_fpoly
M1 M1
M2 M2
```

```
M3 M3
M4 M4
M5 M5
M6 M6
M7 M7
M8 M8
M9 M9
M10 M10
metal11 AP

via_layers
CO n_polyCont
VIA1   VIA1
VIA2   VIA2
VIA3   VIA3
VIA4   VIA4
VIA5   VIA5
VIA6   VIA6
VIA7   VIA7
VIA8   VIA8
VIA9   VIA9
via10  RV
```

SEE ALSO

report_starrc_in_design(2)
check_starrc_in_design(2)

set_starrc_options

Loads and validates a configuration file for running starrc signoff extraction with the StarRC tool used in PT ECO fusion flow.

SYNTAX

status **set_starrc_options**
-config *config_file*

Data Types

config_file string

ARGUMENTS

-config *config_file*

Specifies the name of the configuration file. The configuration file contains settings needed to run the StarRC. The syntax is described in the DESCRIPTION section.

DESCRIPTION

This command loads and validates a configuration file for running PT ECO fusion with the StarRC tool. User needs to specify StarRC command file. The configuration file recognizes asterisk (*) as a comment character and supports the following keywords:

- SIGNOFF_IMAGE : The location of the StarXtract executable
- CORNER_GRD_FILE : The file that lists each corner and the corresponding StarRC nxtgrd file
- MAPPING_FILE : The StarRC layer mapping file
- COMMAND_FILE : The StarRC command file

EXAMPLES

prompt> **set_starrc_options -config file1.cfg**
prompt> **sh cat file1.cfg**

* file1.cfg

```
SIGNOFF_IMAGE:  /global/apps/bin/StarXtract
MAPPING_FILE:   ./tcad/nxtgrd.mapping
CORNER_GRD_FILE: ./tcad/cornerGrd.mapping
COMMAND_FILE:   starRC.cmd
```

SEE ALSO

report_starrc_options(2)

set_stub_chain

Sets the specified stub chain in the current design.

SYNTAX

status **set_stub_chain**
[-at *index*]
[-remove |
[-remove_element_with_pin pin |
[-remove_all |
[-add_floating_elements *element_lists* |
[-add_ordered_elements *element_lists]*
stub_chain

Data Types

index int
elements_lists list

ARGUMENTS

-at *index*

Position within the *stub_chain* to add/remove element_lists.

-remove

Remove the element list at the specified position in the *stub_chain*.

-remove_element_with_pin pin

Remove the single element associated with the specified *pin*.

-remove_all

Remove all element_lists from the *stub_chain*.

-add_floating_elements *element_lists*

Add floating element lists at the specified positon within the *stub_chain*. If position is not specified, list is added at the end of the *stub_chain*. The scan-in and scan-out pin of each triplet must be from the same scan cell. List Format: {{{scan_in_pin_name scan_out_pin_name bit_length}...}...}

-add_ordered_elements *element_lists*

Add ordered element lists at the specified positon within the *stub_chain*. If position is not specified, list is added at the end of the *stub_chain*. The scan-in and scan-out pin of each triplet must be from the same scan cell. List Format: {{{scan_in_pin_name scan_out_pin_name bit_length}...}...}

stub_chain

Name or collection of the stub_chain to be updated.

DESCRIPTION

The **set_stub_chain** command updates the specified stub_chain in the current design. *index* must be specified with -remove, -add_floating_elements and -add_ordered_elements. Option -remove_all removes all the element_lists and doesn't require *index*. Option -remove_element_with_pin can only be specified with pin. For adding element_lists, *index* range can be from 0 to length of stub_chain, for removing element_lists, *index* range can be from 0 to length of stub_chain minus 1.

EXAMPLES

The following example sets floating element_lists at index 1 on a stub_chain named stub1

```
prompt> set_stub_chain -at 1 -add_floating_elements {{{u1/pin1 u1/pin1 4} {u1/pin3 u1/pin4 4}}} stub1
1
```

The following example removes element_lists at index 2 from a stub_chain named stub1.

```
prompt> set_stub_chain -at 2 -remove stub1
```

SEE ALSO

create_stub_chain(2)
get_stub_chains(2)
remove_stub_chains(2)
report_stub_chains(2)

set_supernet_exceptions

Sets the supernet transparency for cells and pins in the design.

SYNTAX

status **set_supernet_exceptions**
-pins *pins_list* | -disable_cells *cells_list*

Data Types

pins_list list
cells_list list

ARGUMENTS

IP "*pins_list*" Specifies the pins in the design which are to be marked as transparent for supernets. The pins in the list must belong to the same instance.

cells_list

Specifies the cells in the design whose supernet transparent settings are to be removed. If the specified cell is a repeater then it is marked as being non-transparent. If the specified cell is non-repeater then supernet transparent settings on its pins are removed.

DESCRIPTION

This command marks cells and pins as being transparent for supernets from the design.

EXAMPLES

The following example marks the pins A1 and Z of the cell cntrl/U166 as transparent to supernets.

prompt> **set_supernet_exceptions -pins {cntrl/U166/A1 cntrl/U166/Z}**

The following example disables the repeater cell i_clock7. The cell is marked as non-transparent to supernets.

prompt> **set_supernet_exceptions -disable_cells {i_clock7}**

SEE ALSO

remove_supernet_exceptions(2)
report_supernet_exceptions(2)

set_supply_net_probability

Annotates switching probability on selected supply nets in the current design.

SYNTAX

status **set_supply_net_probability**
-static_probability *static_probability*
[-scenarios *scenario_list*]
[-modes *mode_list*]
[-corners *corner_list*]
object_list

Data Types

static_probability float
scenario_list list
mode_list list
corner_list list
object_list list

ARGUMENTS

-static_probability *static_probability*

Specifies the value of the *static_probability* of the switching activity of a supply net. The *static_probability* value represents the percentage of time the supply net is in the 'on' state. For a power net the 'on' state is the logic state '1', for a ground net the 'on' state is the logic state '0'. For example, a *static_probability* of 0.25 indicates that the supply net is in the 'on' state 25% of the time.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Probability is set separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, probability is set for the current scenario in the design.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering on modes will occur.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If this option is not specified, no filtering on corners will occur.

object_list

Specified a supply net in the current design for which the value of static probability are set.

DESCRIPTION

This command annotates design supply nets with different kinds of switching activity. The activity of a supply net is used to scale leakage power and dynamic power of the cells connected to the supply net. The scaling of leakage power is governed by app option *power.scale_leakage_power_at_power_off* (by default true), the scaling of dynamic power by app option *power.scale_dynamic_power_at_power_off* (by default false). The scaling factor for a cells is the lowest static_probability of all its supply nets.

The supply nets that are annotated with switching activity can be specified explicitly as a list of objects.

If a combination of a mode and corner is specified that does not represent a valid scenario, those are ignored.

Multicorner-Multimode Support

EXAMPLES

The following **set_supply_net_probability** command annotates a static_probability of 0.9 on a supply net

```
prompt> set_supply_net_probability -static_probability 0.9 [get_supply_net VSS]
```

SEE ALSO

get_supply_net_probability(2)
reset_supply_net_probability(2)
power.scale_leakage_power_at_power_off(3)
power.scale_dynamic_power_at_power_off(3)

set_svf

Specifies the file into which the Formality setup information is written.

SYNTAX

status **set_svf**
[*file_name*]
[-append]
[-off]

Data Types

file_name string

ARGUMENTS

file_name

Names the file into which Formality setup information will be recorded. You must specify the file name unless the **-off** option is specified.

-append

Appends to the specified file. If another Formality setup verification file is already open then it will be closed before opening the specified file. If -append is not used then set_svf will overwrite the named file, if it exists.

-off

Closes the SVF files and stops recording Formality setup information. If this option is specified before any setup information has been recorded, it prevents the default SVF file from being created. No other options can be combined with **-off**.

DESCRIPTION

This command specifies the file name that should be used for recording setup information for Formality, the Synopsys formal verification tool.

If the **set_svf** command is not issued, the tool will create a file automatically, using a date-stamped, host name, and pid name like "default-20200210_dcamd66_1522.svf". An informational message will be printed when the default file is created. You can prevent the default file from being created by issuing the **set_svf -off** command at the beginning of the session, before any setup information is recorded.

The SVF ("Setup Verification for Formality") file that is produced contains Formality "guide" commands, like "guide_environment"

and "guide_uniquify", that are used by Formality during the matching step to facilitate the alignment of compare points. The tool might create additional files in the SVF file to pass checkpoint or DesignWare implementation information.

The SVF file is used to account for changes to the hierarchy or to netlist object names that occur during synthesis. Certain operations performed by the compile command perform transformations that are also recorded in the SVF file.

Further information on SVF files can be found in the Formality User Guide.

Because the SVF output is buffered internally, the SVF files might not be complete until recording is stopped. You can stop recording by running **set_svf** *-off* or by exiting from the tool. Running **set_svf** with a new file name will write out and close the original SVF file before starting the new one in the new file.

The SVF information is stored in encrypted format.

To record setup information for formal verification products other than Formality, use the **set_vsdc** command, instead.

This command returns a Boolean value indicating whether **set_svf** succeeded (true) or not (false).

EXAMPLES

In this example, we query the name of the SVF file and find out that setup information is not being recorded yet. Then we specify the file name and repeat the query.

```
prompt> get_svf
SVF guidance is not enabled.
1
prompt> set_svf cache_ctrl_svf
1
prompt> get_svf
The current SVF file is /project/chip/cache_ctrl/cache_ctrl_svf.
1
```

SEE ALSO

get_svf(2)
set_vsdc(2)
Formality User Guide

set_switching_activity

Annotates switching activity on selected nets, pins, ports and cells in the current design.

SYNTAX

status **set_switching_activity**
-static_probability *static_probability*
-toggle_rate *toggle_rate_value*
[-state_condition *condition_name*]
[-path_sources *path_sources*]
[-period *period_value*]
[-base_clock *clock*]
[-scenarios *scenario_list*]
[-modes *mode_list*]
[-corners *corner_list*]
[-clock_derate *clock_derate_value*]
object_list

Data Types

static_probability float
toggle_rate_value float
condition_name string
path_sources list
period_value float
clock string
scenario_list list
mode_list list
corner_list list
clock_derate_value float
object_list list

ARGUMENTS

-static_probability *static_probability*

Specifies the value of the *static_probability* in the switching activity. The *static_probability* value represents the percentage of time the signal is at the logic state '1'. For example, a *static_probability* of 0.25 indicates that the signal is in the logic state '1' for 25% of the time.

-toggle_rate *toggle_rate_value*

Specifies the value of the toggle rate in the switching activity. The rate is a floating point number that represents the number of 0->1 and 1->0 glitch free transitions that the signal makes during a period of time. You can specify the period with the *-period*

option. Alternatively, you can annotate a related clock using the *-base_clock* option, and the specified toggle rate is relative to the related clock period.

-state_condition *condition_name*

Specifies the state condition when annotating state-dependent activities. State dependent activities can be annotated when the internal power of the library cell pin is characterized with state-dependent power tables. The state condition specified with this argument must match to a state condition in the internal power characterization. Use **-state_condition** default to specify the default state condition.

-path_sources *path_sources*

Specifies the path sources when annotating path-dependent activities. This can be used when the library cell pin has path-dependent internal power characterization. The path sources specified with this argument must be the same as those in the internal power characterization. If **-path_sources** option is used to specify the path_sources value, then **-state_condition** option must be specified to specify the value of the path_sources.

-period *period_value*

Specifies the time period for which the number of transitions given in the *toggle_rate_value* occur. The time units for this value are those specified in the main technology library. When this argument is used, the value of toggle rate specified by the *-toggle_rate* option is divided by the given *period_value*. The resulting toggle rate is then annotated to the objects provided in the **set_switching_activity** command. If the *-period* option is not specified, a default *period_value* of '1.0' is assumed.

-base_clock *clock*

Specifies a clock to which toggle rate value are related to. This clock is referred to as the object's related clock. When this option is used, the value of toggle rate specified by the *-toggle_rate* option is divided by the clock period. The resulting toggle rate is then annotated to the objects provided in the **set_switching_activity** command. When the *-base_clock* value is set to "*", the toggle rate is divided by the period of the fastest related clock, if fastest related clock is not found then fastest design clock is used.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. Activity is set separately for each scenario. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, the activity is set for the current scenario in the design.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*. If this option is not specified, no filtering on modes will occur.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*. If this option is not specified, no filtering on corners will occur.

-clock_derate *clock_derate_value*

Can be used as an alternative for **-toggle_rate** option. If specified, the toggle_rate_value used will be the toggle_rate_value of the fastest related clock of the object multiplied by the clock_derate_value.

object_list

Specifies a list of nets, pins, ports or cells in the current design on which the *static_probability* or/and *toggle_rate* values of switching activity are to be set. In case of cells, the probability values can be set for the state_conditions which are already present in the internal library characterization for the corresponding library cells of the cells.

DESCRIPTION

Use this command to annotate design nets, ports, pins and cells with the different kinds of switching activity. These include toggle rate and static probability on nets, ports and pins, along with state_condition based probability values on cells.

You can specify a related clock using the *-base_clock* option. The annotated toggle rate is divided by the related clock period and the resultant toggle rate value is annotated on the design objects. Only one of the *-period* or *-base_clock* option can be specified.

The design objects that are annotated with switching activity can be specified explicitly as a list of objects.

If a combination of a mode and corner is specified that does not represent a valid scenario, those are ignored.

For statistics on the switching activity annotation on the current design, use the **report_switching_activity** command.

It is to be noted that activity information is persistent given that the application option 'power.enable_activity_persistency' is set to 'on' or 'lazy'. Default value of 'power.enable_activity_persistency' is 'lazy'.

Multicorner-Multimode Support

EXAMPLES

The following **set_switching_activity** command annotates a toggle rate value of '0.9' and static probability value of '0.8' on a net.

```
prompt> set_switching_activity -static_probability 0.8 -toggle_rate 0.9 {Product[21]}
```

The following example annotates a toggle rate value of '0.6' and static probability value of '0.9' for the input port 'MemWriteBus[26]'. The clock 'clock' is specified as the base clock for the input port.

```
prompt> set_switching_activity MemWriteBus[26] -static_probability 0.9 \
        -toggle_rate 0.6 -base_clock "clock"
```

The toggle rate value of '0.6' is divided by the clock period of '3.0'. The resulting toggle rate value of '0.2' is annotated on the input port.

SEE ALSO

get_switching_activity(2)
read_saif(2)
report_switching_activity(2)
reset_switching_activity(2)
power.enable_activity_persistency(3)

set_synlib_dont_use

Disables the use of given synthetic implementations or modules during the compilation of the current design.

SYNTAX

status **set_synlib_dont_use** *obj*

Data Types

obj string

ARGUMENTS

obj

Specifies the affected synthetic modules or implementations. The specified modules or implementations must be in a synthetic library already loaded into the tool.

The format of this argument is */library/module/implementation*. The *implementation* is optional. A wildcard pattern can be used. If more than one pattern is included, they must be enclosed in quotation marks or braces ({}).

DESCRIPTION

Directs the tool not to use the specified synthetic modules or implementations during compile. The setting is stored as an attribute in the current design, and can be removed by **remove_attribute** command with attribute name **synlib_dont_use**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command specifies that the *cla* and *rpl* implementation of *DW_div* module in the *dw_foundation* and *standard* library should never be used by the tool:

prompt> **set_synlib_dont_use {dw_foundation/DW_div/cla standard/DW_div/rpl}**

The following command specifies that all *rtl* implementations of all synthetic modules should never be used by the tool:

```
prompt> set_synlib_dont_use */*/rtl
```

SEE ALSO

get_attribute(2)
remove_attribute(2)

set_tap_package_model

Creates package models by setting the parasitic resistance, capacitance, and inductance values to the tap points or the package SPICE models.

SYNTAX

status **set_tap_package_model**
[-power]
[-ground]
[-R *value*]
[-L *value*]
[-C *value*]
[-type *type_name*]

Data Types

float *value*
string *type_name*
list *tap_list*

ARGUMENTS

-power

Model parameters is for power.

-ground

Model parameters is for ground.

-R *value*

Assigns the specified resistance value to the taps.

-L *value*

Assigns the specified inductance value to the taps.

-C *value*

Assigns the specified capacitance value to the taps.

-type *type_name*

Specifies the tap package model when two or more parasitic elements are applied to the taps. The valid *typename* values are

package, *wirebond*, or *pad*.

DESCRIPTION

The **set_tap_package_model** command provides a number of built-in electrical models that are effective in modeling many package configurations. You can select these built-in models by using the -type *type_name* option with the -power, -ground, -R,-L, and -C options to specify the parasitic values of the various model components.

The -R,-L, and-C options are used to define the resistance, inductance, and capacitance values for the model components. When an -R,-L, and -C option is not specified, by default the value is set to 0, meaning the corresponding model component is not used.

When modeling a single electrical component between the tap and the external ideal voltage, specify either of -the R,-L, and -C options. In this case, specifying the -type option is not required.

EXAMPLE

The following example sets a resistance value of 5 units (in SI units) on all power pads. point:

prompt> **set_tap_package_model -R 5 -type pad -power**

The following example sets a resistance value of 0.5 units, and an inductance value of 5 units (in SI units) on the package:

prompt> **set_tap_package_model -type package -R 0.5 -L 5**

SEE ALSO

analyze_rail(2)
set_tap_package_model(2)
rail.database(3)
rail.redhawk_path(3)

set_target_library_subset

Sets target library subsets on the top block and hierarchical cells.

SYNTAX

status **set_target_library_subset**
[-objects *objects*]
[-top]
[-data | -clock]
[-merge none | inherited | current]
[-dont_use *lib_cells*]
[-use *lib_cells*]
[-only_here *lib_cells*]
[*target_libs*]

Data Types

objects	list or collection
lib_cells	list or collection
target_libs	list or collection

ARGUMENTS

-objects *objects*

Specifies the hierarchical cells on which to set the target library subset. The cells must be in the current block.

For each hierarchical cell, the subset restriction applies to the cell for optimization and CTS. The subset also applies to the cell's children, if they are not explicitly assigned their own target library subsets. The cell must reference a hierarchy within the current block. It cannot reference another block.

If you do not specify this option or the **-top** option, the tool uses the **-top** option by default. Specifying both options is also supported.

-top

Sets the target library subset on the top block. The subset also applies to all cells in the hierarchy, if they are not explicitly assigned their own target library subsets.

If you do not specify this option or the **-objects** option, the tool uses the **-top** option by default. Specifying both options is also supported.

-data | -clock

Indicates whether the target library subset applies only to the data paths or clock paths. If not specified, the target library subset applies to all paths. The -data and -clock options are mutually exclusive, you may specify only one.

-merge none | inherited | current

Specifies how the new subset is to be merged with the existing subset, if any, on each of the specified hierarchies.

If the merge style is set to **none**, no merging will occur. The new subset overwrites the existing subset on each hierarchy. This is the default setting when no merge style is specified.

If the merge style is set to **inherited**, the new subset is merged with the inherited subset on each hierarchy. The inherited subset is the one on the parent hierarchy of the hierarchy.

If the merge style is set to **current**, the new subset is merged with the current subset on each hierarchy. The current subset may be inherited from above or previously assigned to the hierarchy.

-dont_use *lib_cells*

Specifies the library cells that cannot be used for optimization and CTS within the specified objects. This is in addition to the restrictions imposed by the *target_libs* option.

The library cells must exist in the reference library list. You can specify library cells by name, such as "mylib/AN2", or by collections, such as "get_lib_cells */AN2". You can also use wildcards. Wildcards are expanded according to the library cells in memory at the time the command is issued.

The **-dont_use** and **-use** options are mutually exclusive; you can use only one.

You must specify at least one of **-dont_use**, **-use**, **-only_here**, or *target_libs*.

-use *lib_cells*

Specifies the library cells that can be used for optimization and CTS within the specified objects. This is in addition to the libraries listed in the *target_libs* option.

The library cells must exist in the reference library list. The syntax for the **-use** and **-dont_use** options are identical.

If a library cell's dont_use attribute is true, it is not used here even if it is listed in the **-use** option. You could use the **get_attribute** and **set_attribute** commands to check and modify the library cell's dont_use attribute.

The set of usable library cells is not stored, instead the complementary set is calculated and stored as the set of library cells that cannot be used. This is equivalent to specifying the **-dont_use** option with the complementary library cell set. So to check this setting, check the attributes and reports pertaining to the **-dont_use** option.

The **-dont_use** and **-use** options are mutually exclusive; you can use only one.

You must specify at least one of **-dont_use**, **-use**, **-only_here**, or *target_libs*.

-only_here *lib_cells*

Specifies the library cells that can be used for optimization and CTS within the specified objects, but cannot be used in other objects unless they also list the library cell in an **-only_here** option. The library cells must exist in the reference library list.

The syntax for the **-only_here** and **-dont_use** options are identical. If a library cell is listed in both, the **-only_here** option overrides the **-dont_use** option.

If a library cell's dont_use attribute is true, it is not used here even if it is listed in the **-only_here** option. You could use the **get_attribute** and **set_attribute** commands to check and modify the library cell's dont_use attribute.

You must specify at least one of **-dont_use**, **-use**, **-only_here**, or *target_libs*.

target_libs

Specifies the libraries that are available for optimization and CTS within the specified objects. The libraries must be a subset of the reference library list.

If you do not specify this option, all reference libraries can be used.

You must specify at least one of **-dont_use**, **-use**, **-only_here**, or *target_libs*.

DESCRIPTION

This command sets the target library subset on the top block and hierarchical cells. Target library subsets restrict optimization and CTS on the specified objects to a subset of reference libraries and library cells.

Optimization is constrained by the target library subsets on the top block and hierarchical cells. Usually during optimization, a library cell is selected from the reference libraries. The set_target_library_subset command allows you to specify or filter the library cells to be used for cells within the top block or specified objects. The subset restriction applies only to cells that are created or modified during optimization. The subset does not affect any unoptimized portions of the block.

When target library subsets are specified on multiple ancestor hierarchies for a given cell, precedence is given to the subset on the hierarchy nearest to the cell. A subset on the lower level completely overrides a subset specified at a higher level.

Use the **remove_target_library_subset** command to remove the target library subset from a top block or hierarchical cell.

Use the **report_target_library_subset** command to report the target library subset of a top block or hierarchical cell.

As with most block constraints, the subset specifications are tied to the current block. If you change the current block to another block, and you want the subset to affect optimization at that level, you must reapply the **set_target_library_subset** command.

EXAMPLES

The following example sets a target library subset on the top hierarchy.

```
prompt> set_target_library_subset -top \
  -only_here [get_lib_cells lib1/AND*] \
  -dont_use  [get_lib_cells lib1/INV*] \
  [get_libs lib1]
```

The following example sets a target library subset on a hierarchical cell.

```
prompt> set_target_library_subset -objects [get_cells top/mid] \
  -only_here [get_lib_cells lib2/AND*] \
  -dont_use  [get_lib_cells lib2/INV*] \
  [get_libs lib2]
```

SEE ALSO

report_target_library_subset(2)
remove_target_library_subset(2)

set_technology

Apply technology node specific settings to current design

SYNTAX

status **set_technology**
-node node_number
-report_only

Data Types

node_number string

ARGUMENTS

-node node_number

Specifies the technology node, such as "7" or "12". User has to specify one of the supported technology node names.

-report_only

Report the list of public app options without applying them.

DESCRIPTION

This command will apply technology specific settings to current design. It sets the parameters for placement and route engines to satisfy advanced node design rules. It also sets some app options to control the tool work correctly. Use -report_only to list the app option names and values to be set. Once the command is executed, all the tecnology node specific controls will be applied to the design, the "technology_node" attribute will be set to the design as well.

It is recommended to use set_technology at the beginning of design flow, e.g. after design is loaded and linked, prior to place and route. The settings are persistent and saved in design database, so that user don't have to call this command multiple times in their flow. However, when user switch to a newer release, it is suggested to use set_technology again just in case the internal settings may be upgraded in the newer release.

EXAMPLES

In the following example, we apply technology node specific settings to a design using 7nm technology.

```
prompt> set_technology -node 7
technology_node=7
1
```

SEE ALSO

set_temperature

Sets the operating temperature for one or more corners, for part or all of the current design.

SYNTAX

status **set_temperature**
temperature
[-min *temperature*]
[-corners *corner_list*]
[-object_list *object_list*]
[-clear]

Data Types

object_list collection
corner_list collection
temperature float

ARGUMENTS

temperature

The temperature to be used.

-min *temperature*

The early temperature, if different than the late temperature.

-corners *corner_list*

The corner(s) to apply the temperature to. If this option is not given, the current corner is used.

-object_list *object_list*

The cell or libcell object(s) to apply the temperature. If this option is not given, the temperature is applied to the entire current design.

-clear

Remove all existing temperature settings on the given corners. The main library nominal temperatures will be used. If this option is given, the **temperature**, **-min temperature**, and **-object_list** options cannot be given.

DESCRIPTION

The **set_temperature** command is used set the operating temperature that will be used for timing analysis and delay calculation.

Multicorner-Multimode Support

By default, this command works on the current corner. To specify a different corner, use the **-corners** option.

EXAMPLES

This example shows how to change the temperature for an entire design, for the current corner.

```
prompt> set_temperature 125.0 -min -40.0
```

This example shows how to change the temperature for an entire design, for all corners.

```
prompt> set_temperature 125.0 -min -40.0 -corner [all_corners]
```

SEE ALSO

set_voltage(2)
set_process_label(2)
set_process_number(2)
report_pvt(2)

set_test_assume

Sets the **test_assume** attribute to a logic value to be assumed on specified cell output pins throughout test design rule checking.

SYNTAX

integer **set_test_assume**
zero_or_one_value
pin_list

Data Types

zero_or_one_value	string
pin_list	list

ARGUMENTS

zero_or_one_value

Specifies the fixed value of the output pins. To assume a constant value of logic one, *zero_or_one_value* must be "1", "one", or"ONE." To assume a constant value of logic zero, *zero_or_one_value* must be "0","zero", or"ZERO."

pin_list

Lists the hierarchical pins in the current design for which values are specified. If more than one pin is specified, they must be enclosed in braces ({ }).

DESCRIPTION

The **set_test_assume** command sets the **test_assume** attribute to *zero_or_one_value* on the cell output pins on *pin_list*. The **dft_drc** command uses **test_assume** to freeze the assumed value on the specified pins throughout test design rule checking.

Use **set_test_assume** to set conditions that would otherwise be unknown to **dft_drc**, such as the state of a non-scan register, or the output of a RAM. Known values are often applied to the design circuitry to facilitate testing by configuring the design circuitry into the desired test mode. For example, during testing, a state could be preloaded into the non-scan registers of an on-chip test controller. This condition could configure the scan chains in the design to form multiple parallel scan chains. Or, if it is known that prior to testing, all address locations of an on-chip RAM are preloaded with a fixed pattern, this pattern can be assumed at the data output pins of the RAM when testing the surrounding functional logic. You can communicate these preloaded values to the testing software through the **test_assume** attribute.

Note: The specified conditions must exist throughout testing, regardless of other changes in the design state. Otherwise, incorrect design rule checking reports can result. If the assumed state is not attainable or not held throughout the scan test sequence, it is

considered best practice to use the initialization protocol to provide initialization vectors that perform the assumed setup, instead of using **set_test_assume**, which could result in errors.

Use the **report_dft -test_assume** command to list the **test_assume** attributes on a design.

To remove selected or all **test_assume** attributes from a design, use the **remove_attributes** or **reset_design** command.

EXAMPLES

The following commands specify a fixed internal state to assume for a register in an on-chip test controller:

```
prompt> set_test_assume 0 my_tap/reg3/Q

prompt> set_test_assume 1 my_tap/reg3/QN
```

The following command specifies that all the data outputs of a RAM cell are fixed throughout testing:

```
prompt> set_test_assume ONE [get_pins h1/ramcell/DOUT*]
```

SEE ALSO

current_design(2)
dft_drc(2)
get_attribute(2)
remove_attributes(2)
report_dft(2)
reset_design(2)
set_dft_signal(2)

set_test_point_element

Configures the insertion of force, control, and observe user-defined test points.

SYNTAX

int **set_test_point_element**
list_of_design_objects
[-type control_0 | control_1
| force_0 | force_1 | force_01
| observe]
[-control_signal *control_name*]
[-clock_signal *clock_name*]
[-test_points_per_source_or_sink *n*]

Data Types

list_of_design_objects	list
control_name	string
clock_name	string
n	integer

ARGUMENTS

list_of_design_objects

Specifies the list of design pin or port objects at which the user-defined test point elements will be inserted. Wildcards and collections are supported. This is a required argument.

-type control_0 | **control_1** | **force_0** | **force_1** | **force_01** | **observe**

Specifies the type of test point to insert at each design object in the list. Specify only one type with each invocation of this command.

-control_signal ***control_name***

Specifies the name of the existing pin or port control signal that activates the test point. The signal must have a type of TestMode, ScanEnable, or IbistEnable.

-clock_signal ***clock_name***

Specifies the name of the clock signal used to clock any inserted user-defined test point scan registers. If no clock signal is specified, the tool determines the clock from the surrounding logic.

In all flows, you can specify the name of a scan clock signal that is predefined with the **set_dft_signal -type ScanClock** command.

In a DFT-inserted OCC controller flow, you can specify the name of a PLL output pin that is predefined with the **set_dft_signal -type Oscillator** command. This specification is mapped to the corresponding OCC controller clock during DFT insertion.

In an existing OCC controller flow, you can directly specify the name of an output pin of an existing OCC controller that is predefined with the **set_dft_signal -type Oscillator** command.

-test_points_per_source_or_sink ***n***

Specifies the number of test points that can be shared per source data signal (for control and force test points) or sink observed signal (for observe test points). The source signal can be a source data scan register or a source signal input port. The sink signal can be an observe scan cell or a sink output port.

A range of 1 to 32 can be specified for *n*.

By default, 8 observe test points can share one sink signal and 8 control or force test points can share one source signal.

DESCRIPTION

The **set_test_point_element** command configures the insertion of force, control, and observe user-defined test points. These test points are subsequently inserted by the **compile_fusion** command.

To use this feature, you must enable the **testability** DFT client:

prompt> **set_dft_configuration -testability enable**

This command requires a list of pins or ports where the test points should be inserted.

Force test points are used when a value needs to be forced throughout the entire test session. A force test point is activated when the control signal (normally the TestMode signal) is asserted. The force_0 and force_1 test point types allow a signal to be replaced with a constant 0 or constant 1 value throughout the entire test session. These test point types are useful when a particular signal must be forced to a known value for testability purposes. The force_01 test point type allows a signal to be replaced with a source data value (driven from a scan register, input port, or user-supplied source data signal) throughout the entire test session.

Control test points are used when a hard-to-control signal should be controllable (selectively forced) for some test vectors, but left unaltered for others. This allows the test program to select either the original signal behavior, or the forced behavior, on a vector-by-vector basis. Control test points are typically inserted to increase the fault coverage of the design. A control test point is like a "selectively-activated" force control point. It works the same way, except that an additional test point enable signal, supplied by a scan register, input port, or user-supplied test point enable signal, must also be asserted.

Observe test points are typically inserted at hard-to-observe signals in a design to reduce test data volume or to increase the coverage. An observe test point passively observes a signal, and captures its value in an observe sink data scan register, or a sink data output port. Multiple observed points within a single observe test point definition can share the same sink point by collapsing the observed signals with an XOR observability tree.

Each **set_test_point_element** command describes a unique test point element definition. Signal sharing is not performed between test point element definitions. If test points within a limited geographic region should share the same source, sink, or enable signals, they should all be provided in a single **set_test_point_element** command. If test points across a wide geographic region should not share signals to avoid routing congestion, they should be broken up into localized groups and specified with separate **set_test_point_element** commands.

EXAMPLES

The following example specifies that observe test points be inserted at the specified set of four design pins. Since **-scan_source_or_sink** is enabled by default, **insert_dft** inserts two new observe scan registers to observe the four pins. The first three pins share one observe scan register, and the last pin is observed by a second observe scan register.

```
prompt> set_test_point_element -type observe {U0/Q U1/Q U2/Q U3/Q} \
    -test_points_per_source_or_sink 3 \
    -clock_signal CLK
```

The following example forces the RSTN pin of an analog block to zero during the entire test program, to hold the analog block in a quiet, low-power reset state:

```
prompt> set_test_point_element -type force_0 {U_ANALOG/RSTN}
```

SEE ALSO

preview_dft(2)

set_testability_configuration

Configures automatic test-point insertion and global test-point insertion parameters.

SYNTAX

status **set_testability_configuration**
[-target *target_type*]
[-test_point_file *file_name*]
[-only_from_file false | true]
[-clock_signal *clock_name*]
[-control_signal *control_name*]
[-test_points_per_scan_cell *n*]
[-include_elements *object_list*]
[-include_fanin_cone *object_list*]
[-include_fanout_cone *object_list*]
[-exclude_elements *object_list*]
[-exclude_fanin_cone *object_list*]
[-exclude_fanout_cone *object_list*]
[-effort low | medium | high]
[-isolate_elements *cell_list*]
[-max_test_points *n*]
[-random_pattern_count *n*]
[-target_test_coverage *coverage_value*]
[-sg_command_file *file_name*]

Data Types

clock_name string
control_name string
n integer
object_list list
cell_list list
coverage_value float
file_name string
threshold_value integer

ARGUMENTS

-target *target_type*

Enables and configures an automatic test point insertion target.

Test points are not inserted for a target until it is enabled by this option. Target lists are supported.

The valid target values are:

- **random_resistant** - inserts **control_0**, **control_1**, and **observe** test points to improve the testability of hard-to-test (random-pattern-resistant) logic.
- **untestable_logic** - inserts **force_01** and **observe** test points to to control uncontrollable nets and observe unobservable logic, respectively.
- **x_blocking** - inserts **force_01** test points at unknown-function output pins, such as the output pins of black boxes.
- **multicycle_paths** - inserts capture-blocking logic at endpoints constrained by multicycle paths.
- **shadow_wrapper** - inserts a shadow wrapper around each cell specified with the **-isolate_elements** option (**observe** test points at input pins, **force_01** test points at output pins).
- **core_wrapper** - inserts test-point-based core wrapping in the design (**force_01** test points at input ports, **observe** test points at output ports).
- **user** - inserts user-supplied test points described in the external file defined by the **-test_point_file** option

If the **-target** option is omitted, the command can only configure the following global (non-target-specific) options:

- **-test_point_file**
- **-clock_signal**
- **-control_signal**
- **-test_points_per_scan_cell**
- **-sg_command_file**

-test_point_file ***file_name***

Specifies a file that describes test points to insert.

Each line contains three whitespace-separated fields: a user-defined label (can be any string), the test point type keyword, and the net where the test point should be inserted. Any text after the '#' comment character is ignored, whether in-line or on a separate line.

If a test point comes from a SpyGlass run (with an in-line SpyGlass comment), it belongs to that target. Otherwise, it belongs to the **user** target.

Note that file-based test points are not implemented unless the target is enabled (just as with analysis-based test points). For more information, see the DESCRIPTION section.

Supported scopes: global

-only_from_file false | **true**

Specifies whether file-based test points should augment or replace analysis-based test points.

When set to **false** (the default) for a target, the tool performs analysis for it, then implements both analysis-based and file-based test points for it.

When set to **true** for a target, the tool does not perform analysis for it; it implements only file-based test points for it. Any options that affect analysis for that target will have no effect.

This option is valid only when the **-test_point_file** option is also set. For more information, see the DESCRIPTION section.

Supported scopes: all targets

-clock_signal ***clock_name***

Specifies the name of a previously defined ScanClock signal to use for DFT-inserted test point registers.

The default is to use the dominant clock of the logic surrounding each test point.

Supported scopes: global, all targets

-control_signal *control_name*

Specifies the name of control signal to use for enabling test points. The signal must have a type of TestMode or lbistEnable.

The default is to use an available TestMode signal.

Supported scopes: global, all targets

-test_points_per_scan_cell *n*

Specifies the number of test points that can be shared by a single test-point register.

The default is 8.

Supported scopes: global, all targets

-include_elements *object_list*

Specifies hierarchical cells that should be fixed. Wildcards and collections are supported.

The default is to perform fixing across the entire design.

Supported scopes: **random_resistant**, **untestable_logic**

-include_fanin_cone *object_list*

Specifies ports or pins whose fanin logic should be fixed. Wildcards and collections are supported.

The default is to perform fixing across the entire design.

Supported scopes: **random_resistant**, **untestable_logic**

-include_fanout_cone *object_list*

Specifies ports or pins whose fanout logic should be fixed. Wildcards and collections are supported.

The default is to perform fixing across the entire design.

Supported scopes: **random_resistant**, **untestable_logic**

-exclude_elements *object_list*

Specifies objects that should be excluded from fixing. Wildcards and collections are supported.

The default is not to exclude any objects from fixing.

Supported scopes: **random_resistant** (hierarchical cells), **untestable_logic** (hierarchical cells), **core_wrapper** (ports)

-exclude_fanin_cone *object_list*

Specifies ports or pins whose fanin logic should be excluded from fixing. Wildcards and collections are supported.

The default is not to exclude any objects from fixing.

Supported scopes: **random_resistant**, **untestable_logic**

-exclude_fanout_cone *object_list*

Specifies ports or pins whose fanout logic should be excluded from fixing. Wildcards and collections are supported.

The default is not to exclude any objects from fixing.

Supported scopes: **random_resistant**, **untestable_logic**

-effort low | **medium** | **high**

Specifies the effort level used to compute the optimal test points for hard-to-test logic. Higher effort levels yield more efficient test points, but at the expense of analysis runtime.

The default is **medium**.

Supported scopes: **random_resistant**

-isolate_elements ***cell_list***

Specifies hierarchical, black-box, or CTL-modeled cells to be isolated with a test-point shadow wrapper. Wildcards and collections are supported.

The default is an empty list.

Supported scopes: **shadow_wrapper**

-max_test_points ***n***

Specifies the maximum number of test points to be inserted.

This limit applies to the **random_resistant** and **untestable_logic** targets. When applied globally, the tool allocates the limit across both targets. You can also apply per-target limits, which takes precedence.

The default is 5% of the register count of the design.

Supported scopes: global, **random_resistant**, **untestable_logic**

-random_pattern_count ***n***

Specifies the number of random patterns to statistically model when performing random-resistant analysis. Coverage values specified with the **-target_test_coverage** option are interpreted according to this value.

The default is 8000.

Supported scopes: **random_resistant**

-target_test_coverage ***coverage_value***

Specifies the random-pattern target test coverage in percentage points that, when reached, prevents further test points from being generated.

The default is 100.

Supported scopes: **random_resistant**

-reuse_threshold ***threshold_value***

Specifies the maximum number of I/O registers that can be reused as shared wrapper cells when wrapping the design with test points.

The default is 0.

Supported scopes: **core_wrapper**

-depth_threshold ***threshold_value***

Specifies the maximum combinational depth of I/O registers that can be reused as shared wrapper cells when wrapping the design with test points.

The default is 1.

Supported scopes: **core_wrapper**

-sg_command_file ***file_name***

Specifies a file of SpyGlass Tcl commands file to be applied to the test-point analysis.

This options file is an unordered list of SpyGlass Tcl commands that you want to apply. The tool automatically applies the Tcl commands at the proper point in the analysis based on their type (design read, setup, goal definitions, etc.).

Supported scopes: global

DESCRIPTION

The **set_testability_configuration** command enables and configures automatic test-point insertion for one or more targets.

To use this feature, you must first enable the testability DFT client:

```
prompt> set_dft_configuration -testability enable
```

Issue this command without the **-target** option to configure global test-point insertion parameters. These global options are noted above.

Issue this command with the **-target** option to configure the specified target(s). Each target supports a particular subset of this command's options, described above. Warning messages are issued if you specify unsupported options for a target.

Specify the global configuration first, followed by the per-target configurations.

The following global options can also be specified per-target, which takes precedence over their global value:

- **-clock_signal**
- **-control_signal**
- **-test_points_per_scan_cell**

By default, analysis-based test points are implemented for a target when it is enabled. If you specify the **-test_point_file** option, any file-based test points are also implemented for that target. To suppress analysis and implement only the file-based test points, set the **-only_from_file** option to **true** for that target.

EXAMPLES

The following example configures three automatic test-point insertion targets:

```
prompt> # global settings
prompt> set_testability_configuration \
        -test_points_per_scan_cell 20 \
        -control_signal TM_TP
```

```
prompt> # target-specific settings
prompt> set_testability_configuration -target random_resistant \
        -random_pattern_count 1000 -target_test_coverage 95

prompt> set_testability_configuration -target {x_blocking core_wrapper}
```

The following example configures insertion of only file-based test points for two targets from a previous SpyGlass run, without rerunning analysis:

```
prompt> # global settings
prompt> set_testability_configuration \
        -control_signal TM_TP \
        -test_point_file spyglass.txt

prompt> # enable targets for file-based insertion only
prompt> set_testability_configuration -only_from_file true \
        -target {random_resistant x_blocking}

prompt> # specifying exclude elements as name
prompt> set_testability_configuration -target random_resistant -exclude_elements { sa_eeprom } \

prompt> # specifying exclude elements as collection
prompt> set_testability_configuration -target random_resistant -exclude_elements [ list $all_cells ]  \

prompt> # specifying include_elements as list of collection, names and name with wildcard
prompt> set_testability_configuration -target random_resistant -include_elements [ list $all_cells sa_int_ctrl sa_reg_access
```

SEE ALSO

report_testability_configuration(2)
reset_testability_configuration(2)
run_test_point_analysis(2)
set_dft_configuration(2)

set_threshold_voltage_group_type

This command specifies which threshold voltage group names for library cells are high, normal, or low vt type.

SYNTAX

string **set_threshold_voltage_group_type**
  -type *vth_type*
  group_names

string *vth_type*
list *group_names*

ARGUMENTS

-type *vth_type*

Specifies the threshold voltage type for the specified groups. Must be one of **high_vt**, **normal_vt**, or **low_vt**.

group_names

A list of threshold voltage group names, which are stored as **threshold_voltage_group** attributes on lib_cell objects.

DESCRIPTION

This command specifies which threshold voltage group names for library cells are high, normal, or low vt type.

It is to be noted that threshold voltage group type information is persistent given that the app option 'power.enable_activity_persistency' is set to 'on'.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example specifies that lib_cells with "threshold_voltage_group" attribute equal to "HVT" or "hvt" are considered as "high_vt" type for optimization.

```
prompt> set_threshold_voltage_group_type -type high_vt {HVT hvt}
```

SEE ALSO

set_attribute(2)
set_max_lvth_percentage(2)
power.enable_activity_persistency(3)

set_timing_derate

Sets the delay derating factors for either the current design or a specified list of cells or library cells.

SYNTAX

int **set_timing_derate**
-early | -late
[-rise]
[-fall]
[-clock]
[-data]
[-cell_delay]
[-cell_check]
[-net_delay]
[-static]
[-dynamic]
[-variation]
[-aocvm_guardband | -pocvm_guardband]
[-pocvm_subtract_sigma_factor_from_nominal]
[-pocvm_coefficient_scale_factor]
[-increment]
[-corners *corner_list*]
derate_value
object_list

Data Types

corner_list list
derate_value float
object_list list

ARGUMENTS

-early

Applies the *derate_value* to early delays (shortest paths). This option and the **-late** option are mutually exclusive.

-late

Applies the *derate_value* to late delays (longest paths). This option and the **-early** option are mutually exclusive.

-rise

Applies *derate_value* to rise delays.

-fall

Applies *derate_value* to fall delays.

-clock

Applies the specified derating factor only to clock paths.

-data

Applies the specified derating factors only to data paths.

-cell_delay

Applies the specified derating factor to cell delays. This option cannot be specified together with the **-cell_check** option.

-cell_check

Applies the specified derating factor only to cell timing checks. If this option is specified, then the derate value set with the **-early** option is applied to hold and removal timing checks, and the derate value set with the **-late** option is applied to setup and recovery timing checks. This option cannot be specified together with the **-clock** and **-data** options. This option cannot be specified with the **-cell_delay** and **-net_delay** options.

-net_delay

Applies the specified derating factor only to net delays. This option cannot be specified together with the **-cell_check** option.

-static

Applies the specified derating factor only to the non-delta portion of net delays. This option also requires that you specify the **-net_delay** option.

-dynamic

Applies the specified derating factor only to the dynamic component of delays. This option also requires that you specify the **-net_delay** option.

-variation

Applies the specified derating factor to statistical delays only. This option is currently ignored.

-aocvm_guardband

Applies a delay to a timing arc that is a product of the guardband derate factor and the advanced OCV derate factor. This option is only applicable in an advanced on-chip variation (AOCV) context. This option cannot be specified together with the **-cell_check** option.

-pocvm_guardband

This option is only applicable in a parametric on-chip variation (POCV) context. In a parametric OCV context, the derate factor that is applied to an arc is a product of the guard-band derate factor and the parametric OCV derate factor corresponding to a distance-based parametric OCV table, if specified. The parametric OCV factor applied to the standard deviation is also multiplied by the guardband factor.

-pocvm_subtract_sigma_factor_from_nominal

This option is only applicable in a parametric on-chip variation (POCV) context and requires constraint variation to be enabled. The derate factor is for subtracting the intrinsic standard deviation from the nominal value for constraint arcs. This option can be used only with the **-cell_check** option, which applies the derating to cell timing check constraints. Note that this derate factor is applied prior to **-pocvm_guardband** option and **-pocvm_coefficient_scale_factor** option.

-pocvm_coefficient_scale_factor

This option is only applicable in a parametric on-chip variation (POCV) context. In a parametric OCV context, the coefficient applied to an arc is a product of the parametric on-chip variation (POCV) coefficient and the parametric on-chip variation (POCV) coefficient scale factor.

-increment

Specifies incremental derate factor that is added to the base derate factor to compute the total derate for an object. This option can not be specified with the options, **-aocvm_guardband**, **-pocvm_guardband** or **-pocvm_coefficient_scale_factor**.

-corners *corner_list*

Specifies the corners that the derate value applies to. If this option is not given, the current corner is used.

derate_value

Specifies the timing derate value to apply to the specified delays as a scalar multiplying factor.

object_list

Specifies the current design, or list of cells, library cells, or nets to which the specified derate factor is applied. If no objects are given, the derate factor is applied to the entire design.

DESCRIPTION

This command sets derating factors for either the current design (if no object list is specified) or a set of cells or library cells in the current design.

Timing derating factors affect delay values shown in timing reports. Longest path delays (for example, launching paths for setup checks or capturing paths for hold checks) are multiplied by the derate value set by using the **-late** option. Shortest paths (such as capturing paths for setup checks or launching paths for hold checks) are multiplied by the derate values set by using the **-early** option. If derating factors are not specified, a value of 1.0 is assumed.

Begin by first setting global or default derates on the design. Then, you can use the *object_list* option to set specific derate values for cells or library cells in the design. To set derate values globally on a design, issue this command with no object list specified.

For example, to set an early derate factor of value 0.95 on all cell and net delays in the design, use the following command:

```
set_timing_derate -early 0.95
```

If only the **-net_delay** option is specified, and no *object_list* is specified, then the derate factor supplied is applied to all net delays in the design.

If only the **-cell_delay** option is specified, then the derate factor supplied is applied to either all cell delays in the design (if no *object_list* is specified) or to each delay (apart from timing check delays) on each cell or library cell specified in the *object_list*.

If neither **-net_delay**, **-cell_delay** nor *object_list* are specified with the command, then it is assumed that the derate factor applies to all cell and net delays in the design.

If only the **-clock** option is specified, then the derate factor supplied is only applied to delays that are in clock paths. If only the **-data** option is specified, then the derate factor supplied applies only to delays that are in data paths. If neither of the above options is specified, then it is assumed that the derate factor applies to both clock and data paths.

The **-cell_check** option is the only option not assumed by default. If specified, this option applies the derate factor to either all timing checks in the design (if no *object_list* is specified) or to a specific list of cells or library cells (specified by *object_list*). Note that the following commands sets all derating factors except for constraints:

```
set_timing_derate -late 1.3
```

set_timing_derate -early 0.8

To set derating for constraints, use the following command

set_timing_derate -late 1.3 -cell_check
set_timing_derate -early 0.8 -cell_check

The incremental derate value, specified using the **-increment** option, is applied to both simple derates as well as AOCVM and POCVM derates. For a given object, the base derate is first computed and then the incremental derate value is added to determine the total derate factor. The incremental derate factors follow the same inheritance, precedence and override rules as the base derates. If the total derate factor after addition of incremental derate value is negative, the tool will limit it to 0.0 and a warning message is issued.

With SI enabled, net delays might also include delta delay resulting from crosstalk interaction between nets. Users can specify separate derate factors for static and dynamic(delta delay) net delays by using **-static** and **-dynamic** options. The net derate factors are applied separately to the static and dynamic delays first and then both derated values are summed to give the total derated net delay.

When the options, **-static** and **-dynamic**, are not specified for net delays, both static and dynamic factors are set. Both -static and -dynamic are supported with -increment option. With aocvm and pocvm, the derate factors from aocvm/pocvm are used along with corresponding static or dynamic incremental derate factors. When old format constraints, where dynamic derates were not supported, are loaded into versions with -static/-dynamic support, both static and dynamic values are set. This is same behavior as set_timing_derate with no -static/-dynamic specified.

The *object_list* option can be used to set specific derate factors on instances (cells or library cells) in the design. On each instance, the options **-cell_delay**, **-cell_check**, **-clock** and **-data** can be used in the same fashion as outlined above to specify exactly how the derate factors should be applied to each instance in the object list.

If no options are specified and an object list is given, then it is assumed that **-cell_delay**, **-clock** and **-data** are TRUE.

When applying derate factors, the tool uses the following priority, starting from the highest:

1. Derate factors set on the leaf cell
2. Derate factors set on the library cell
3. Derate factors set on the hierarchical cell
4. Derate factors set on the design

To set derating values to the default (derate factor of 1.0 for every instance in the design), use the **reset_timing_derate** command.

For delays, an early factor less than 1 and late factor more than 1 adds pessimism to the slack calculation. However, constraint derating is applied directly to the value of constraint. For example, a factor of less than 1 for hold decreases the hold time and gives a less conservative slack. A factor greater than 1 for setup makes setup time larger and gives a more conservative slack.

The following equation is used to apply derates to both positive and negative delays.

derated_delay = delay + ((derate_factor - 1.0)*ABS(delay))

This equation ensures that negative delays will be correctly made more conservative by the application of derates.

To report all derate factors set for the design or for any cell, library cell or net in the design, use the **report_timing_derate** command.

Multicorner-Multimode Support

By default, this command works on the current corner. To specify a different corner, use the **-corners** option.

EXAMPLES

The following example sets an early (shortest path) derate factor of 0.7 and a late (longest path) derate factor of 1.0 globally on all cell and net delays the design.

```
prompt> set_timing_derate -early 0.70
prompt> set_timing_derate -late 1.0
```

The following example sets an early derate factor of 0.8 for the cell delay of cell U1.

```
prompt> set_timing_derate -early 0.8 -cell_delay [get_cells U1]
```

The following example sets a late derate factor of 1.1 on all instances of library cell IV in the library MY_LIB.

```
prompt> set_timing_derate -late 1.1 [get_lib_cells MY_LIB/IV]
```

Assuming that cell, "inv" is a particular instantiation of MY_LIB/IV in the design, the following command overwrites the late derate factor for the instance "inv" only.

```
prompt> set_timing_derate -late 1.05 [get_cells inv]
```

The following example sets a derate factor on all cells within the hierarchy top/H1.

```
prompt> set_timing_derate -cell_delay -late 1.05 [get_cells top/H1]
```

The following example uses incremental derates to set a late derate factor of 1.15 (= 1.11 + 0.04) and early derate factor of 0.88 (= 0.91 + (-0.03)) for the cell delay of cell top/H1/u12.

```
prompt> set_timing_derate -late 1.09
prompt> set_timing_derate -early 0.91
prompt> set_timing_derate -late 1.11 [get_cells top/H1/u12]

prompt> set_timing_derate -increment -late 0.04 [get_cells top/H1/u12]
prompt> set_timing_derate -increment -early -0.03 [get_cells top/H1/u12]
```

The following example sets static and dynamic derates for net delays to 1.11 for late and 0.88 for early.

```
prompt> set_timing_derate -late 1.1 -net_delay
prompt> set_timing_derate -early 0.88 -net_delay
```

The following example sets different static and dynamic late derates for net delays.

```
prompt> set_timing_derate -late 1.1 -net_delay -static
prompt> set_timing_derate -late 0.28 -net_delay -dynamic
```

The following example sets total static and dynamic derates of 1.2 with incremental derates for late net delays.

```
prompt> set_timing_derate -late 1.07 -net_delay -static
prompt> set_timing_derate -late 0.13 -net_delay -static -increment
prompt> set_timing_derate -late 1.18 -net_delay -dynamic
prompt> set_timing_derate -late 0.02 -net_delay -dynamic -increment
```

The following example sets -pocvm_guardBand, -pocvm_coefficient_scale_factor and -pocvm_subtract_sigma_factor_from_nominal of 1.1, 0.8, and 0.2 for late constraint arc. Assume that intrinsic constraint arc delay in POCV analysis is (nominal, sigma) and POCV corner sigma is 3.0. After applying above derating factors, the derated constraint arc delay will be ((nominal - 0.2 * sigma) * 1.1, sigma * 1.1 * 0.8 * 3.0);

```
prompt> set_timing_derate -late -cell_check -pocvm_guardband 1.1
prompt> set_timing_derate -late -cell_check -pocvm_coefficient_scal_factor 0.8
prompt> set_timing_derate -late -pocvm_subtract_sigma_factor_from_nominal 0.2
```

The following command sets an early derating factor of 0.75 on a specific net, n123, overriding the global derating factor for net delays:

```
prompt> set_timing_derate -net_delay -early 0.75 [get_nets n123]
```

SEE ALSO

report_design(2)
report_timing(2)
report_timing_derate(2)
reset_timing_derate(2)
set_operating_conditions(2)

set_timing_paths_disabled_blocks

Ignore the internal reg2reg timing paths across subblocks.

SYNTAX

string **set_timing_paths_disabled_blocks**
[-all_sub_blocks]

list *blocks_list*

ARGUMENTS

-all_sub_blocks

When specified, all reg2reg timing paths across subblocks will be ignored.

DESCRIPTION

This command controls whether tool needs to ignore the internal reg2reg timing paths across subblocks.

Sometimes, during top level optimization with subblocks, user might only focus on the top and boundary logic.

With this command, user could ignore block level reg2reg timing paths which has worst timing violations.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

prompt> **set_timing_paths_disabled_blocks -all_sub_blocks**

SEE ALSO

remove_timing_paths_disabled_blocks(2)
report_timing(2)
report_qor(2)

set_top_module

Specifies which module to use as top module in the current unlinked block, and links it

SYNTAX

status **set_top_module**
[*module_name*]

Data Types

module_name string

ARGUMENTS

module

Specifies the name of the module to use as top module while linking the current block. The current block must be unlinked.

DESCRIPTION

This command performs a name-based resolution of block references for the current block. The specified module is used as the top module. References are resolved using other modules within this block, top modules of other physical hierarchy blocks, and leaf cells within lib cell libraries.

Auto Link

Many commands automatically link the current block before executing. These automatic links are always incremental, and might not fully rebind block references when the reference library list or view switch list is changed. To force a full relink and rebinding of the block, use the **link_block -force -rebind** command.

EXAMPLES

In the following example, block TOP in library blockLib is linked to the physical hierarchy blocks MID and BOT, and the lib cell library leafCellLib.

```
prompt> set_top_module TOP
Using libraries: designLib leafCellLib
```

```
Linking design TOP
Linking design BOT
Linking design MID
Design 'TOP' was successfully linked.
1
```

SEE ALSO

link_block(2)

set_topology_edge_shapes

Sets the shape layers for selected or specified topology edges in the current design.

SYNTAX

status **set_topology_edge_shapes**
[-shape l_x_first | l_y_first | l_longer_first | z_x_first | z_y_first | z_longer_first]
[-force]
[*object_list*]

Data Types

object_list collection or selection

ARGUMENTS

-shape l_x_first | l_y_first | l_longer_first | z_x_first | z_y_first | z_longer_firs

Specifies the shape layers template for topology edge.

The shape values are:

l_x_first is L-shape with first segment in x direction
l_y_first is L-shape with first segment in y direction
l_longer_first is L-shape with first segment in longer direction
z_x_first is Z-shape with first segment in x direction
z_y_first is Z-shape with first segment in y direction
z_longer_first is Z-shape with first segment in longer direction

-force

Sets shapes for locked or fixed objects. By default, such objects are skipped according to global edit settings. Edit settings can be accessed using **get_edit_setting**.

DESCRIPTION

The **set_topology_edge_shapes** command sets or resets shape layers for selected or specified topology edges.

Note if option **-shape** is not specified the shape template will be chosen automatically with respect to the access direction of the start and end topology nodes.

EXAMPLES

The following example sets L-shape layers with first segment in x direction for selected topology edges.

prompt> **set_topology_edge_shapes -shape l_x_first**

The alternative syntax uses collection to specify objects.

prompt> **set_topology_edge_shapes [get_selection] -shape l_x_first**

SEE ALSO

create_topology_edge(2)
get_topology_edges(2)
get_edit_setting(2)
get_selection(2)
set_edit_setting(2)

set_trace_option

Set an option value controlling behavior of command tracing and output annotation..

SYNTAX

set_trace_option [-command name
|-profile profile_metric_types
|-memory_threshold threshold
|-cpu_threshold threshold
|-time_treshold threshold]
[-annotate annnotate_type]

string *name*
string *annotate_type*
list rofile_metric_types
integer *threshold*

ARGUMENTS

-command *name*

This option is mutually exclusive with -profile, -memory_threshold, -cpu_threshold, and -time_threshold. The option identifies the command for which annotation option is being set.

-annotate *annotate_type*

This option is meaningful only when given with the -command option. Set the annotation type of the given command with this option. The allowed values are {annotate, omit}.

The *annotate* option value causes the traced command execution to be annotated to the shell output. If the application supports an output log, the annotation is also output to the output log.

The *omit* option value causes the command execution string to be omitted from output annotation.

-profile *profile_metric_types*

This option is mutually exclusive with all others. The option sets the currently enabled profile metrics, or "all" if all metrics are enabled. Allowed values are a list of: memory, cpu, time, or the value all to indicate enabling of all metrics, or the value none to indicate disabling of all metrics. If none is given in a list with any other value, it is ignored. The enabled profile metrics are reported with command annotations in the tool output to shell, and to the output log if supported by the tool.

-memory_threshold *threshold*

This option is mutually exclusive with all others. The option sets the threshold for gathering memory metrics. The profile data annotation will be output at the end of a command inovcation only when the delta memory use for a command invocation

matches or exceeds the threshold. The threshold has no effect on profile data output at the beginning of command invocation when this is enabled. The threshold is expressed in kilobyges. The special value -1 unsets the threshold.

-cpu_threshold *threshold*

This option is mutually exclusive with all others. The option sets the threshold for gathering CPU usage metrics. The profile data annotation will be output at the end of a command inovcation only when the delta CPU usage for a command invocation matches or exceeds the threshold. The threshold has no effect on profile data output at the beginning of command invocation when this is enabled. The threshold is expressed in seconds. The special value -1 unsets the threshold. If cpu threshold isunset, then the default threshold of 1 second is used. To effect no threshold, set the threshold to 0 (zero).

-time_threshold *threshold*

This option is mutually exclusive with all others. The option sets the threshold for gathering elapsed time metrics. The profile data annotation will be output at the end of a command inovcation only when the elapsed time for a command invocation matches or exceeds the threshold. The threshold has no effect on profile data output at the beginning of command invocation when this is enabled. The threshold is expressed in seconds. The special value -1 unsets the threshold. If time threshold is unset, then the threshold for cpu is used.

DESCRIPTION

If a command is given with -annotate option, then the **set_trace_option** command informs the tool on how to behave during command annotation output, which is started with the **annotate_trace** command.

If profile, memory_threshold, cpu_threshold, or time_threshold is given, the the **set_trace_option** configures the behavior of performance profile gathering and the data reported in command annotations to the output.

EXAMPLES

The following example sets profile configurations, turns annotation off for the command *get_attribute*, then starts command annotations to the output.

```
prompt> set_trace_topion -profile all
prompt> set_trace_topion -memory_threshold 10
prompt> set_trace_topion -cpu_threshold 30
prompt> set_trace_topion -command get_attribute -annotate omit
prompt> annotate_trace -start -profile on
```

SEE ALSO

log_trace(2)
annotate_trace(2)
get_trace_option(2)

set_track_constraint

The command defines a set of track constraints on per-layer basis. The constraints describe rules for the structure of track objects in the block to ensure that the generate wire tracks meet manufacturability criteria

SYNTAX

int **set_track_constraint**
[-block *block*]
-grid *grid*
-label *label*
-layer *layer*
[-mask *masks*]
-offset *offset*
-origin block | core_area
[-track_direction horizontal | vertical]

Data Types

block string
grid int
label string
layer string
masks list of masks
offset list of float

ARGUMENTS

-block *block*

The -block argument specifies the block on which to set the track constraints.

-grid *grid*

The -grid argument specifies the length of the window for the allowable track offset pattern.

-label *label*

The -label argument specifies the label assigned to the track constraint definition generated by the command. Labels must be unique, but may apply to muliple track constraint classifications if more than one is covered in the constraints definition.

-layer *layer*

The -layer argument specifies the layer for which the track constraints apply. Only valid interconnect layers may be given.

-mask *masks*

The -mask argument specifies the mask(s) for which the constraint applies. Mask values currently supported are "mask_one" and "mask_two". The constraint will apply only to tracks that exist on the given mask(s). Giving a list with both "mask_one" and "mask_two" means the constraint will apply to tracks on either mask_one or mask_two.

Omitting the -mask argument will assign the constraint to un-colored tracks.

-offset *offset*

The -offset argument specifies the allowable positions for tracks relative to a multiple of the constraint window (grid) value. The list should contain only non-negative floats in increasing order. The list should not contain any values greater than the value given for the -grid argument. The list may contain only one number but should not be empty.

-origin block | core_area

The -origin argument specifies the reference point for the track constraint pattern defined. A value of "block" means the constraint pattern starts from the boundary of the block. A value of "core_area" means the constraint pattern starts from the boundary of the block's core area.

-track_direction horizontal | vertical

The -track_direction argument specifies the track orientation for which the constraint applies. A value of "horizontal" means the constraint applies only to tracks parallel to the X axis. A value of "vertical" means the constraint applies only to tracks parallel to the Y axis. Omitting this argument means the constraint will apply to both horizontal and vertical tracks.

DESCRIPTION

The set_track_constraint command defines valid wire track structures in the given block on a per-layer basis. The constraint itself is comprised of an origin, grid, and offset. The origin describes the reference point from which to start the pattern. The constraint pattern itself is defined as each offset value starting from each multiple of the grid value. In other words, the constraint pattern says that tracks are valid if they exist at a location given by the expression O + n*G + F for any F in the offset list.

The defined constraint applies to the tracks specified by the block, layer, track_direction, and mask arguments.

Lastly, the label serves as an identifier for the given constraint. Labels may not be re-used once they have been applied. The same constraint may, however, be overwritten with a different label. Constraints (and their corresponding labels) may be removed using the remove_track_constraint command. Once a label has been removed (which happend by removing all constraints covered by that label) it may be re-assigned to a new set of constraints.

EXAMPLES

The following command creates a constraint that applies to both vertical and horizontal tracks with a mask color of 1 on layer M1 in block current_block. The constraint says that valid tracks can exist at offset values of 0, 0.2, 0.8, 1, 1.2, 1.8, 2, etc. from the boundary of the block.

```
prompt> set_track_constraint -block [current_block] -layer M1 \
  -label A -grid 1 -offset {0 0.2 0.8} -origin block -mask mask_one
```

SEE ALSO

remove_track_constraint(2)
report_track_constraints(2)

set_ungroup

Either prevent ungrouping or ungroup cells during **compile**.

SYNTAX

int **set_ungroup**
object_list true | false

Data Types

object_list list

ARGUMENTS

object_list

Specifies a list of cells or modules. Specifying modules is same as specifying all instances of modules.

true | **false**

The value which determines weather to ungroup cells or to prevent their ungrouping. True ungroups cells. False prevents ungrouping. Default is **true**.

DESCRIPTION

set_ungroup is used to either prevent ungrouping of cells or force ungrouping of cells during **compile**. Cells are prevented from ungrouping if **set_ungroup** is specified with false switch. Cells are ungrouped if **set_ungroup** is specified with true switch. Ungrouping of cells is done early in the optimization so that optimization engine sees ungrouped hierarchies.

Specifying modules is same as specifying all instances of modules.

The **compile** command does not ungroup cells with other constraints such as those marked using **set_dont_touch**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, cell U1 is ungrouped during **compile**.

prompt> **set_ungroup [get_cells U1] true**

In the following example, cell U1 is prevented from ungrouping during **compile**.

prompt> **set_ungroup [get_cells U1] false**

In the following example, all instances of module M are prevented from ungrouping during **compile**.

prompt> **set_ungroup [get_modules M] false**

In the following example, all instances of module M are ungrouped during **compile** .

prompt> **set_ungroup [get_modules M] true**

SEE ALSO

compile(2)
ungroup(2)

set_units

This command has no effect. It exists for backwards compatibility when reading SDC files with the **read_sdc** command.

SYNTAX

int **set_units**
[-time *[optional float][optional scale_value]s*]
[-capacitance *[optional float][optional scale_value]F*]
[-resistance *[optional float][optional scale_value]Ohm*]
[-voltage *[optional float][optional scale_value]V*]
[-current *[optional float][optional scale_value]A*]
[-power *[optional float][optional scale_value]W*]

character *scale_value [f|p|n|u|m|k|M]*

ARGUMENTS

-time ***optional floatoptional scale_values***

-capacitance ***optional floatoptional scale_valueF***

-resistance ***optional floatoptional scale_valueOhm***

-voltage ***optional floatoptional scale_valueV***

-current ***optional floatoptional scale_valueA***

-power ***optional floatoptional scale_valueW***

DESCRIPTION

This command has no effect. It exists for backwards compatibility when reading SDC files with the **read_sdc** command. **read_sdc** will parse the **set_units** command, but since the command has no effect, the **set_user_units** may need to be executed before **read_sdc** if the SDC file's units are not the same as the current user input units. **write_sdc** and **write_script** will write the **set_units** command as the first command at their output, based on the output units currently in effect.

The units used by commands and reports can be set at any time with the **set_user_units** command. All input and output units can be reported by **report_units** command.

SEE ALSO

set_user_units(2)
report_units(2)
read_sdc(2)
write_sdc(2)
write_script(2)

set_unix_variable

This is a synonym for the **setenv** command.

SEE ALSO

gettenv(2)
printenv(2)
printvar(2)
set(2)
setenv(2)
sh(2)
unset(2)

set_unloaded_register_removal

Sets the **remove_unloaded_register** attribute on the specified cells or modules, allowing removing of the unloaded registers.

SYNTAX

status **set_unloaded_register_removal**
obj_list
[true | false]

Data Types

obj_list list

ARGUMENTS

obj_list

Specifies a list of cell or module names for which to remove unloaded registers. Cell names in *obj_list* must be from the current design. If multiple objects are specified, they must be enclosed in quotation marks ("") or braces ({}).

true | false

Specifies the value with which to set the **remove_unloaded_register** attribute. When this option is set to true (the default), unloaded register removal is enabled on the specified objects (cells or modules). Unloaded register removal is performed on the specified objects provided that the application option **compile.seqmap.remove_unloaded_registers** is set to true.

A setting of false on a module do not remove unloaded registers for the whole module. A setting of false on an instance do not remove unloaded registers for the whole instance.

DESCRIPTION

The **set_unloaded_register_removal** command sets the **remove_unloaded_register** attribute on the specified *obj_list*. This attribute is used to specify that the cells or modules are to be remove unloaded registers.

If a cell with a specified name is found in the current design, the **remove_unloaded_register** attribute is set to the specified value on the cell. If no cell with a specified name is found, the tool searches for a module and the attribute set on the module.

EXAMPLES

The following example shows how to enable register removal during optimization for the sequential cells named U0 and U1 and how to disable register removal for the U2 cell:

```
prompt> set_unloaded_register_removal {U0 U1} true
prompt> set_unloaded_register_removal U2 false
```

SEE ALSO

compile(2)
get_attribute(2)

set_user_units

This command sets the units that are used for user input and/or output.

SYNTAX

string **set_user_units**
  [-input]
  [-output]
  -type *unit_type*
  -value *unit_value*

string *unit_type*
string *unit_value*

ARGUMENTS

-input

Set the units for data input such as source and read_sdc.

-output

Set the units for data output such as reports and attributes.

-type *unit_type*

Specifies the type of unit to set. This is one of "capacitance", "current", "length", "power", "resistance", "time" or "voltage".

-value *unit_value*

Specifies the unit value. For time, equivalent examples would include "1ps", "1.0ps", "1e-12".

DESCRIPTION

This command sets the units that are used for data input and/or output. The tool maintains separate units for input and input. There are unit values for time, resistance, capacitance, power, voltage and current.

EXAMPLES

The following example sets input and output units to 10ps, 1kOhm, 1pF, 1V and 1mA.

```
prompt> set_user_units -type time -value 10ps
prompt> set_user_units -type resistance -value 1kOhm
prompt> set_user_units -type capacitance -value 1pF
prompt> set_user_units -type voltage -value 1V
prompt> set_user_units -type current -value 1mA
```

The following example sets input time units to 1ps.

```
prompt> set_user_units -input -type time -value 1ps
```

SEE ALSO

get_user_units(2)
report_user_units(2)

set_variation

Applies absolute voltages values of the allowed overshoot(positive threshold) or undershoot(negtive threshold) in the voltage unit of the library.

SYNTAX

status **set_variation**
[-supply *supply_net*]
[-tolerance *tolerance_value*]

Data Types

supply collection
tolerance float

ARGUMENTS

-supply *supply_net*

This option is optional for specifying supply nets.

-tolerance *tolerance_value*

This option is required to specify one tolerance value or two tolerance values. If one only tolerance value specified, it will be applied to both overshoot and undershoot. if two tolerance values specified, the first will be applied to undershoot and the second will be applied to overshoot.

DESCRIPTION

Defines the overshoot and undershoot tolerance for voltage reconciliation on the supply nets and their corresponding netgroup.

EXAMPLES

The following example sets global tolerance value that will be applied to all supplies, only one value specified, both overshoot and undershoot will take the value, means both postive and negative threshold of 0.2 units around the nominal value on all supplies will be considered during voltage reconciliation.

```
prompt> set_variation -tolerance {0.2}
```

The following example sets global tolerance value that will be applied to all supplies. Two values specified as {0.4 0.2}, 0.4 will be set as negative threshold units, 0.2 will be set as positive threshold units.

```
prompt> set_variation -tolerance {0.4 0.2}
```

The following example sets tolerance value for certain supply nets. The value(s) will be applied to all supply nets of the corresponding netgroup.

```
prompt> set_variation -tolerance {0.2} -supply {BLK/VDD}
prompt> set_variation -tolerance {0.4 0.2} -supply {VDDSS}
```

SEE ALSO

connect_supply_net(2)
-reconcile(2)

set_vclp_options

Used to set VCLP related options if users invoke VCLP using check_vclp_design.

SYNTAX

status **set_vclp_options**
[-vclp_exec]
[-work_dir]
[-search_path]
[-link_library]
[-vc_static_home]
[-reset]

Data Types

-vclp_exec string
-work_dir string
-search_path string
-link_library string
-vc_static_home string
-reset boolean

ARGUMENTS

-vclp_exec ***Path to 'vc_static_shell' or the VCLP executable to be used***

Specifies the path to the VCLP executable to be used. The path must be enclosed in double quotes. This argument is optional.

-work_dir ***path to VCLP Working directory***

Specifies the path to the VCLP working directory. This argument is optional. The path must be enclosed in double quotes.

-search_path ***VCLP search path***

Specifies VCLP search path to be used. This argument is optional. The path must be enclosed in double quotes.

-link_library ***VCLP link library***

Specifies VCLP link libraries. This argument is optional. The path must be enclosed in double quotes.

-vc_static_home ***VC_STATIC_HOME environment variable***

This option can be used to set the VC_STATIC_HOME environment variable. This argument is optional. The path must be enclosed in double quotes.

-reset ***Resets all options***

This option can be used to reset all VCLP related options. This argument is optional.

DESCRIPTION

The **set_vclp_options** command is used to set all the VCLP related options is users invoke VCLP in ICC2/FC using **check_vclp_design**.

EXAMPLES

prompt> **set_vclp_options -vc_static_home "/remote/path/example/Release"**

SEE ALSO

check_vclp_design(2)

set_verification_checkpoints

Enables or disables checkpointing within compile.

SYNTAX

status **set_verification_checkpoints**
[-show]
[-off]
[-version *version_name*]
[checkpoints]

Data Types

checkpoints list
version_name string

ARGUMENTS

-show

Lists each checkpoint and whether it is on or off.

-off

Disables all checkpoints. Note that the preRetiming checkpoint is independent of this command and is enabled in the flow if needed.

checkpoints

Specifies a list of checkpoints to enable.

If this argument is provided, then the specified checkpoints will be enabled and all other checkpoints will be disabled. If this argument is not provided, then all supported checkpoints are enabled.

-version *version_name*

Release version supporting compatible ndm lib to use while writing checkpoints. Current default is to use the version of the binary. To find the list of release versions run command report_versions. Allowed values are <version string> and "latest".

DESCRIPTION

This command allows the user to turn on checkpointing within compile. The use of checkpoints can help aid formal verification.

The supported checkpoints are:

ckpt_pre_map ckpt_logic_opt

EXAMPLES

In the following example, the **set_verification_checkpoints** command is used to enable all supported checkpoints.

```
prompt> set_verification_checkpoints
Enabling checkpoint "ckpt_pre_map".
Enabling checkpoint "ckpt_logic_opt".
1
```

In the following example, the **set_verification_checkpoints** commands is used to enable "ckpt_logic_opt" and disable all other supported checkpoints.

```
prompt> set_verification_checkpoints {ckpt_logic_opt}
Disabling checkpoint "ckpt_pre_map".
Enabling checkpoint "ckpt_logic_opt".
1
```

The -show option can be used to display which checkpoints are enabled.

```
prompt> set_verification_checkpoints -show
ckpt_pre_map: off
ckpt_logic_opt: on
Release version: latest
1
```

The -version option can be used to change the library version. Below usage will enable ckpt_pre_map and ckpt_logic_opt checkpoints and also set the version used for writing the library.

```
prompt> set_verification_checkpoints -version O-2018.06
Enabling checkpoint "ckpt_logic_opt".
Enabling checkpoint "ckpt_pre_map".
1
```

To set only the version without enabling ckpt_pre_map and ckpt_logic_opt checkpoits

```
prompt> set_verification_checkpoints -version O-2018.06 -off
Disabling checkpoint "ckpt_logic_opt".
Disabling checkpoint "ckpt_pre_map".
1
```

SEE ALSO

report_versions(2)

set_verification_priority

Sets the **verification_priority** attribute on specified modules or cells.

SYNTAX

status **set_verification_priority**
-all
[-high]
object_list

Data Types

object_list list

ARGUMENTS

-all

Sets the attribute on all modules (equivalent to **[get_modules *]**).

-high

Sets the verification priority to high level. At this level, datapath extraction will be turned off; and high level datapath optimzations are turned off.

object_list

Specifies a list of cells or modules on which the attribute is to be set. If the specified cell is a leaf level cell that is not unmapped DesignWare, the command will skip it.

DESCRIPTION

This command sets the **verification_priority(SVP)** attribute on the specified objects. During **compile**, the optimizations of the design with the attribute are adjusted, some optimization features will be disabled. The disabled optimization features are: ungrouping will be disabled for the cells with SVP, or the cells reference to modules with SVP; Ungrouping of DesignWare components in the design with the attribute is disabled. Some datapath optimizations will be disabled, including extraction of select operators; mutually exclusive resource sharing, etc.

When the attribute is set on a synthetic operator, such as an adder or a multiplier, the resource sharing and the datapath extraction is not performed on the operator. The DesignWare hierarchy of the operator is preserved, except that the size of the DesignWare is small.

The attribute is not transitive to its subdesigns. However, if a design with no SVP attribute is ungrouped into the parent design that has SVP setting, the ungrouped design will honor the parent's SVP setting.

You can use the **remove_verification_priority** command to remove the **verification_priority** attribute.

EXAMPLES

In the following example, the **verification_priority** attribute is set on the *U1* cell:

```
prompt> set_verification_priority [get_cells U1]
```

If the **verification_priority** attribute is specified on a module object, cells using that module are not ungrouped. In this example, all instances of ADDER in the current module are not ungrouped:

```
prompt> set_verification_priority [get_module ADDER]
```

SEE ALSO

compile(2)
remove_verification_priority(2)

set_via_def

Set via def of input vias.

SYNTAX

collection **set_via_def**
[-vias *via_list* | -cells *cell_list*]
[-via_def *via definition*
[-pitch {*horizontal vertical*}]
[-size {*rows columns*}]

Data Types

via_list	collection
cell_list	collection
via definition	string
orientation	string
horizontal	float
vertical	float
rows	integer
columns	integer

ARGUMENTS

-vias *via_list*

Specifies a list of vias to update via definition. This list may contain via collections specified using the get_vias command.

This argument is mutually exclusive with *-cells*.

-cells *cell_list*

Specifies a list of cells to update via definition. This argument only applies to through-silicon via ("TSV") cells, which may be associated with via_defs of type "through_silicon_via_def". TSV cells are identified by their *is_tsv* attribute being true. If any non-TSV cells are specified in this argument, they will be ignored and a warning message issued.

This argument is mutually exclusive with *-vias*.

-via_def *via definition*

Specifies the via definition.

-pitch {*horizontal vertical*}

Specifies the horizontal and vertical pitch values between cuts of a simple array via. The values must be non-negative.

-size {*rows columns*}

Specifies the number of rows and columns (i.e., cut array size) in a simple array via. The values must be 1 or greater. When not specified, the default cut array size is 1x1. When the size is 1x1, simple array via is to change into a simple via.

DESCRIPTION

This command changes via definition of input vias or through-silicon ("TSV") cells.

This command can be used to update simple vias', simple array vias', or custom vias' via definitions. By changing via def and/or size/pitch, a new via with different via definition type might be created. For example, changing via def of a simple via to a custom via def will create a custom via. No custom array via will be created.

This command can also be used to specify the via_def to be associated with TSV cells, which are identified by their *is_tsv* attribute being true. TSV cells may be associated with via_defs of type "through_silicon_via_def", which can be set using this command. TSV cells may also be set to reference user-specifed designs (using the *set_reference* command) with design type "tsv", which is mutually exclusive to referencing a via_def.

EXAMPLES

The following example changes simple via 'VIA_S_1' into a simple array via with via definition 'VIA12'.

```
prompt> set_via_def -vias [get_vias VIA_S_1] -via_def VIA12 -pitch {0.8 0.6} -size {2 18}
```

SEE ALSO

get_vias(2)

set_via_ladder_candidate

Sets via ladder candidates for the specified library-pin.

SYNTAX

status **set_via_ladder_candidate**
lib_pin
-ladder_name via_ladder_name

Data Types

lib_pin pin name or collection
via_ladder_name string

ARGUMENTS

lib_pin

Specifies the library-pin for which via ladder candidate will be set.

-ladder_name *via_ladder_name*

Specifies the via ladder template name present in technology file.

DESCRIPTION

This command sets the via ladder name as via ladder candidate for specified library-pin. This command is additive and should be called once per pair of via ladder candidate and library-pin combination. Each invocation adds another candidate for the specified library-pin. This command returns 1 if succeeded, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the options.

EXAMPLES

The following example set the via ladder candidate for the library-pin.

```
prompt> set_via_ladder_candidate [get_lib_pin tcbn90ghvt/AN2HVTD0/A1] -ladder_name "VL1"
1
```

The following examples add via ladder candidates for the same library-pin.

```
prompt> set_via_ladder_candidate [get_lib_pin tcbn90ghvt/AN2HVTD0/A1] -ladder_name "VL2"
1
prompt> set_via_ladder_candidate [get_lib_pin tcbn90ghvt/AN2HVTD0/A1] -ladder_name "VL3"
1
```

SEE ALSO

reset_via_ladder_candidates(2)
report_via_ladder_candidates(2)
set_via_ladder_constraints(2)
remove_via_ladder_constraints(2)
report_via_ladder_constraints(2)

set_via_ladder_constraints

Sets via ladder constraints for specified pins.

SYNTAX

status **set_via_ladder_constraints**
-pins *pins*
via_ladder_list

Data Types

pins	pin names or collection
via_ladder_list	string

ARGUMENTS

-pins *pins*

Specifies pins for which via ladder constraints will be set. Pins must have block context and should belong to current block.

via_ladder_list

Specifies space-separated ordered-list of via ladder template names present in technology file.

DESCRIPTION

This command sets the list of via ladder names as via ladder constraints for specified pins. If invalid via ladder name is specified, error message is printing out with list of valid template names present in technology file. Constraints set with this command will be read by Zroute to insert via ladders in pins before routing. From the ordered list, first appropriate via ladder will be inserted in the pin.

This command returns 1 if succeeded, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the options.

EXAMPLES

The following examples set the specified constraints for the pins

```
prompt> set_via_ladder_constraints -pins "u1/i1 u2/i2" "VL1 VL2 VL3"
1

prompt> set_via_ladder_constraints -pins [get_pins u2/i3] "VL2 VL3"
1
```

SEE ALSO

remove_via_ladder_constraints(2)
report_via_ladder_candidates(2)
report_via_ladder_constraints(2)
reset_via_ladder_candidates(2)
set_via_ladder_candidate(2)

set_via_ladder_rules

Sets via ladder rules for the block.

SYNTAX

status **set_via_ladder_rules**
-master_pin_map *masterpin_ladders_list*
-master_pin_map_file *file*
-default_ladders *via_ladders_list*
-all_instances_of *cell_master_pin_list*
-all_clock_outputs true | false
-all_clock_inputs true | false
-all_pins_driving *port_list*
-remove_all_rules

Data Types

masterpin_ladders_list string
file string
via_ladder_list string
cell_master_pin_list string
port_list collection

ARGUMENTS

-master_pin_map *masterpin_ladders_list*

Specifies what type of ladder(s) should be used on an instance of given cell_master/pin if it is to get a ladder. Format of masterpin_ladders_list is {{cell_master/pin {via_ladder_list}}...}. The cell_master should be a library cell, entries where cell_master is not a library cell would be ignored with a warning.

-master_pin_map_file *file*

Specifies file name to provide entries for -master_pin_map option. Format of file entry is same as needed in option -master_pin_map. master_pin_map entries can be provided with both the options together. In case of clash, entry in file will be given priority. The cell_master should be a library cell, entries where cell_master is not a library cell would be ignored with a warning.

-default_ladders *via_ladders_list*

Specifies what ladder to use if a cell/pin requires a ladder based on a rule, but it is not defined in the master_pin_map list. Format of via_ladders_list is {VL1 VL2 ...}.

-all_instances_of *cell_master_pin_list*

Specifies that anytime an instance of this cell master/pin is used, a ladder is to be created automatically. Format of cell_master_pin_list is {cell_master1/pin1 cell_master2/pin2 ...}. The cell_master should be a library cell, entries where cell_master is not a library cell would be ignored with a warning.

-all_clock_outputs true | false

Specifies that all output pins on clock nets get a ladder. Default values is false.

-all_clock_inputs true | false

Specifies that all input pins on clock nets get a ladder. Default values is false.

-all_pins_driving *port_list*

Specifies that all pins driving one of the ports in the collection gets a ladder.

-remove_all_rules

If this option is specified, all the rules will be removed.

DESCRIPTION

This command sets via ladder rules for the block. These rules will be used by zroute to automatically update via ladders as the design changes. Instance/pin specifications set with via ladders constraints commands will remain, and can/should still be used for fixed/non-changing cells. These rules are more valuable for dynamically changing cells. If both an instance specific constraint and a rule exist, the instance constraint should apply.

This command returns 1 if succeeded, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the options.

EXAMPLES

The following examples set the specified constraints for the pins

```
prompt> set_via_ladder_rules -all_pins_driving [get_ports Clk] \
  -all_instances_of {mRegister/Ck mAlu/PC[30]}
1
```

SEE ALSO

remove_via_ladder_rules(2)
report_via_ladder_rules(2)
set_via_ladder_constraints(2)
remove_via_ladder_constraints(2)
report_via_ladder_constraints(2)

set_via_ladder_spacing

Specify the placement spacing constraints to accommodate via ladders among a collection of instances or library cells.

SYNTAX

status **set_via_ladder_spacing**
-cells *cell_list*
-lib_cells *lib_cell_list*
-distance *margin_spec*

Data Types

cell_list collection of cell instances
lib_cell_list collection of library cells
margin_spec list

ARGUMENTS

-cells *cell_list*

Specifies a collection of cell instances on which to apply this spacing constraint.

-lib_cells *lib_cell_list*

Specifies the collection of library cells on which to apply this spacing constraint.

-side *margin_spec*

Specifies the spacing requirement in horizontal and vertical dimension in um. Use the following syntax to specify the margins:

{*distance_x distance_y*}

DESCRIPTION

This command adds placement constraint on a collection of library cells or instances which prevent other cells with any such constraint (which can be of different dimension) to be placed within the distance specified by the argument to this command. This rule is satisfied while searching for legal placement.

EXAMPLES

The following example puts via ladder spacing constraint on a collection of 3 instances requiring no other via ladder spacing constrained cell to be placed within 0.3um in horizontal and 0.4um in vertical dimensions.

```
prompt> set_via_ladder_spacing -cells [ get_cells {U1/A U2/D XYZ} ] -distance {0.3 0.4}
```

SEE ALSO

legalize_placement(2)

set_view_switch_list

Sets the value of the switch list for the specified design, library, or global view.

SYNTAX

status **set_view_switch_list**
[-design *design* | -library *library* | -global]
views

Data Types

design	string
library	string
views	list

ARGUMENTS

-design *design*

Specifies the design for which to set the view switch list. This option is mutually exclusive with the **-library** and **-global** options.

-library *library*

Specifies the library for which to set the view switch list. This option is mutually exclusive with the **-design** and **-global** options.

-global

Sets the global, session-wide view switch list. This option is mutually exclusive with the **-design** and **-library** options.

views

Sets the value for the specified entity's view switch list. Specify *views* as a space-delimited list of view names, enclosed within curly brackets. Note that view switch lists are order dependent. Specifying an empty view switch list value clears the view switch list setting on the entity, and the view switch list value becomes the owning library's view switch list if the entity is a design, or the global view switch list if the entity is a library.

DESCRIPTION

This command sets the view switch list of a design, library, or global session.

The view switch list is a precedence-ordered list of design views that is applied when binding a design. If a design has its view switch list explicitly set, then that view switch list is used. If not, then the design's owning library's view switch list is used. If the

owning library does not have a view switch list explicitly set, then the global view switch list is used. The view switch list attribute is persistent for designs and libraries, but not for the global session.

It is an error to specify more than one of the **-global**, **-library**, or **-design** options. If none of these options are specified, the command sets the global view switch list. Use "{}" to specify an empty view switch list. Setting an entity's view switch list to the empty value clears its view switch list setting, resulting in no explicitly set view switch list on the entity. In that case, its effective view switch list becomes that of its owning entity.

EXAMPLES

The following example sets the view switch list of design "mydesign":

prompt> **set_view_switch_list -design mydesign {abstract outline}**

The following example clears the view switch list of design "mydesign":

prompt> **set_view_switch_list -design mydesign {}**

The following example sets the view switch list of library "mylib":

prompt> **set_view_switch_list -library mylib {frame abstract design}**

Each of the following examples have the same effect of setting the global view switch list:

prompt> **set_view_switch_list {design frame abstract outline}**

prompt> **set_view_switch_list -global {design frame abstract outline}**

SEE ALSO

get_view_switch_list(2)
change_view(2)

set_virtual_pad

Creates a virtual power or ground pad as a voltage source for power network analysis.

SYNTAX

status **set_virtual_pad**
-net *net*
-coordinate {*x y*}
[-layer *layer_name*]

Data Types

net collection
x float
y float
layer_name collection

ARGUMENTS

-net *net*

Specifies the net to which to add the power tap. This is a required option.

-coordinate *coordinate*

Specifies the coordinate of the virtual pad. This is a required option.

-layer *layer_name*

Specifies the layer name on which to create the virtual pad.

DESCRIPTION

This command creates a virtual pad for power network analysis.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example adds a virtual pad.

```
prompt> set_virtual_pad -net VDD -coordinate {1033.275 1860.740}
```

The following example adds a virtual pad on the M6 layer.

```
prompt> set_virtual_pad -net VDD -coordinate {1035.000 1890.850} -layer M6
```

SEE ALSO

analyze_power_plan(2)
read_virtual_pad_file(2)
remove_virtual_pads(2)
report_virtual_pads(2)
write_virtual_pad_file(2)

set_voltage

Applies an operating voltage to a list of supply nets, supply ports, cells or ports, or to the current design.

SYNTAX

status **set_voltage**
max_voltage
[-min *min_voltage*]
[-dynamic *dynamic_voltage*]
[-min_dynamic *min_dynamic_voltage*]
[-corners *corner_list*]
[-object_list *objects*]
[-cell *cell*]
[-pg_pin_name *pg_pin*]

Data Types

max_voltage float
min_voltage float
dynamic_voltage float
min_dynamic_voltage float
corner_list collection
objects collection
cell string
pg_pin string

ARGUMENTS

max_voltage

Specifies the operating voltage for the maximum delay (slowest) case. This is typically a smaller voltage value. If the **-min** option is not given, this value will also be used for the minimum delay (fastest) case.

-min *min_voltage*

Specifies the operating voltage for the minimum delay (fastest) case. This is typically a larger voltage value.

-dynamic *dynamic_voltage*

Specifies the dynamic voltage for the maximum delay case. The dynamic portion is considered during pessimism removal. The dynamic component of voltage is provided as a negative value in this case. The static voltage is determined by subtracting the dynamic component from the regular voltage

-min_dynamic *min_dynamic_voltage*

Specifies the dynamic voltage for the minimum delay case.

-corners ***corner_list***

Specifies the corners that will use the given voltages. If this option is not given, the current corner will be used.

-object_list ***supply_objects***

Specifies the supply nets, supply ports, cells, or ports that have the operating voltages specified in this command. If this option is not given, the voltages will be applied to the top level of the current design.

-cell ***cell***

This option exists only for script compatibility, and is ignored.

-pg_pin_name ***pg_pin***

This option exists only for script compatibility, and is ignored.

DESCRIPTION

Defines the operating voltage on the supply nets and their corresponding power nets so that the parts of the design powered by these supply nets or power nets are timed and optimized at the specified voltage. If a voltage is set on a supply net that is part of a connected supply net network, the voltage value will be propagated over the entire network.

Primary voltages may also be set on individual cells or ports, or on the top level of the design. However, the preferred methodology is to assign voltages to the supply nets.

If you do not specify an operating voltage for a supply net, the part of the design connected by this supply net continues to be timed and optimized based on the available operating condition settings.

To see the operating voltages of the supply nets, use the **report_supply_nets** command.

To see the operating voltages of the power pins of a particular cell, use the **report_cells -power** command.

The operating voltage of a power net, once defined, can only be overridden with another **set_voltage** command. It can also be cleared by using the **reset_design** command.

Do not forget that the min (BC) voltage is typically *larger* than the max (WC) voltage!

Multicorner-Multimode Support

By default, this command works on the current corner. To specify a different corner, use the **-corners** option.

EXAMPLES

The following example sets max (worst-case) and min (best-case) voltage values of 1.05 and 1.2 on the supply net VDD1, for the corner corner3:

prompt> **set_voltage 1.05 -min 1.2 -corner corner3 -object_list [get_supply_net VDD1]**

The following example sets the max and min top-level default primary voltage to 1.05 volts for the current corner. This value will be used for any cells that are not powered by supply nets.

```
prompt> set_voltage 1.05
```

SEE ALSO

connect_supply_net(2)
create_power_domain(2)
create_supply_net(2)
report_cells(2)
report_power_domains(2)
report_supply_nets(2)
set_domain_supply_net(2)

set_voltage_area

Updates a voltage_area in the current design.

SYNTAX

int **set_voltage_area**
[-add_power_domains *domain_list*]
[-remove_power_domains *domain_list*]
[-power supply_net]
[-ground supply_net]
[-nwell supply_net]
[-pwell supply_net]
[-add_cells cell_list]
[-remove_cells cell_list]
[-remove_all_cells]
[-is_fixed]
[-merge_regions]
[-name voltage_area_name]
voltage_area_object

Data Types

domain_list name or collection of one or more power_domain objects
supply_net name or collection of a single supply net object
cell_list name or collection of one or more cell objects.
voltage_area_name string
voltage_area_object name or collection of a single voltage_area object.

ARGUMENTS

-add_power_domains *domain_list*

Specifies a list of power domains to be added to this voltage_area. All power domains must be associated with the current design and have the same primary supply net. If a power domain already associated with this voltage_area is specified, it is a no-op and a warning message is printed.

-remove_power_domains *domain_list*

Specifies a list of power domains to be removed from this voltage_area. It is an error if the power domain(s) aren't associated with this voltage_area. If the voltage_area contains explicitly associated cells that belong to the power domains to be removed, the voltage_area becomes invalid. When the last power domain is removed, the use_power_domain_cells attribute is reset to false.

-power

Specifies the power supply net for voltage_area. The net must match one of the available power supply nets on the power

domains. Note that you can only use this option on gas-station type voltage_area.

-ground

Specifies the ground supply net for the voltage_area. The net must match one of the available ground supply nets on the power domains. Note that you can only use this option on gas-station type voltage_area.

-nwell

Specifies the nwell supply net for voltage_area. The net must match one of the available power supply nets on the power domains. Note that you can only use this option on gas-station type voltage_area.

-pwell

Specifies the pwell supply net for voltage_area. The net must match one of the available ground supply nets on the power domains. Note that you can only use this option on gas-station type voltage_area.

-add_cells

Specifies a list of cells to be added to the voltage_area. This option can only be used on gas-station type voltage_area. All cells must be at the same physical hierarchy as the current design, and belong to one of voltage_area's associated power domains. Cells can be physical (leaf) or hierarchical. When specifying hierarchical cells, the cells will be expanded into a set of physical cells that reside under them. If cells are later added or removed from this hierarchical cell, the cell membership in the voltage_area is not automatically updated. Note that one hierarchical cell can expand into multiple physical cells. In (the unlikely) case that the hierarchical cell is purely logical (when it corresponds to a local module that has no physical representation), the cell will map to an empty set of physical cells, and a warning message will be printed. The specified cells cannot already be explicitly or implicitly associated with another voltage_area. When used with the -add_power_domains option, the specified cells must also belong to one of the power domains. If any of the power domains already have an associated voltage_area with implicit cell membership (whose use_power_domain_cells attribute is true) created, the cells will be excluded from the implicit cell list. This is important for ensuring that each cell can associate with only one voltage_area.

-remove_cells

Specifies a list of cells to be removed from the voltage_area. All cells must be already explicitly associated with the voltage_area, not merely implicitly associated with the voltage_area via the -use_power_domain_cells option, or it will be an error. Note that this option can be used along with the -user_power_domain_cells option to remove all explicitly associated cells first, and then creates an implicit cell association using the cell memberships from the associated power domains.

-remove_all_cells

Removes all explicitly added cells from this voltage_area.This option is exclusive with the -remove_cells and -add_cells option. If the voltage_area does not have any explicitly associated cells, this option does nothing, and a warning message will be printed.

-is_fixed

Specifies if the voltage_area is at a fixed location. When using this option, the voltage_area must have at least one shape, or it is an error.

-merge_regions

Merge all abutted and overlapping shapes within this voltage_area into a minimum set of disjoint shapes. The merged shape in each resulting disjoint shape is assigned to the shape with the highest stacking order within the group. The rest of the lower stacking order shapes are removed. This option is only valid when there are at least two shapes in the voltage_area. Note that if a shape has explicit cell assignment, that shape will remain discrete and it will not be merged with other shapes in the same voltage_area even if they touch or overlap. For the same reason, this option cannot be used with the -cells option.

-name

Changes the name of the voltage_area. The name cannot conflict with any existing voltage_areas in this design. Note that the name "DEFAULT_VA" is reserved for the auto-generated default voltage_area, and cannot be used.

DESCRIPTION

This command can update the parameters of a voltage_area. You can add or remove cells (if they are being explicitly specified). You can change the power, ground, and isolation power and ground supply nets. You can specify whether or not the voltage_area is fixed. You can specify or update whether or not the voltage_area is fixed. Finally, you can change the name of the voltage_area.

EXAMPLES

The following example adds a cell "uMid1" to the voltage_area VA1:

prompt> **set_voltage_area VA1 -add_cells uMid1 1**

The following example update the voltage_area to add additional power domains

prompt> **set_voltage_area -add_power_domains {INST1} VA1** 1

prompt> **set_voltage_area -add_power_domains {INST2 INST3} VA2** 1

SEE ALSO

get_voltage_areas(2)
remove_voltage_areas(2)
report_voltage_areas(2)
create_voltage_area_shape(2)
set_voltage_area_shape(2)
remove_voltage_area_shapes(2)
get_voltage_area_shapes(2)

set_voltage_area_shape

Updates a voltage_area shape

SYNTAX

collection **set_voltage_area_shape**
[-add_cells *cell_list*]
[-remove_cells *cell_list*]
[-remove_all_cells]
[-lower | -raise |
-top | -bottom |
-above *voltage_area_shape* | -below *voltage_area_shape*]
voltage_area_shape_object

Data Types

cell_list string or collection
voltage_area_shape string or collection
voltage_area_shape_object string or collection

ARGUMENTS

-add_cells *cell_list*

Specifies a list of cells to be added to this voltage_area shape. All cells must be at the same physical hierarchy as the current design. Cells can be physical (leaf) or hierarchical. When a hierarchical cell is specified, the cells will be expanded into a set of physical cells that reside under them. If cells are later added or removed from this hierarchical cell, the cell membership of the voltage_area shape is not automatically updated. Note that one hierarchical cell can expand into multiple physical cells. In (the unlikely) case that the hierarchical cell is purely logical (when it instantiates to a local module that has no physical representation), the cell will map to an empty set of physical cells, and a warning message will be printed. The specified cells cannot already be explicitly or implicitly associated with another voltage_area or explicitly associated with another voltage_area shape of this same voltage_area.

-remove_cells *cell_list*

Specifies a list of cells to be removed from this voltage_area shape. All cells must be already explicitly associated with this shape, or it will be an error.

-remove_all_cells

Removes all cells from this voltage_area shape. This option is mutually exclusive with the **-remove_cells** and **-add_cells** options. If the voltage_area shape does not have any explicitly associated cells, this option does nothing, and a warning message will be printed.

-lower

Places the voltage_area shape one position lower in the voltage_area shape stack.

-raise

Places the voltage_area shape one position higher in the voltage_area shape stack.

-top

Places the voltage_area shape to the top of the voltage_area shape stack.

-bottom

Places the voltage_area shape to the bottom of the voltage_area shape stack, right above the first shape of the default voltage_area.

-above *voltage_area_shape*

Places the voltage_area shape above the given *voltage_area_shape*.

-below *voltage_area_shape*

Places the voltage_area shape below the given *voltage_area_shape*.

DESCRIPTION

The **set_voltage_area_shape** command updates the parameters of a voltage_area shape. You can add or remove cells (if they are being explicitly specified in the owning voltage_area). You can also raise or lower a voltage_area shape in the shape position stack with the **-raise** and **-lower** options, or bring the voltage_area shape to the top (**-top**) or push it to the bottom (**-bottom**) of the position stack. You can also position it below (**-below**) or above (**-above**) another voltage_area shape in the same design.

The command returns a collection of newly created voltage_area_shape if the command succeeded. Otherwise, the command returns "" if it failed, or TCL_ERROR if there was a command syntax error.

The **-top**, **-bottom**, **-above**, and **-below** options are mutually exclusive. It is a syntax error to specify more than one of these options at a time. It is an error to place a shape above or below a shape from a different design.

EXAMPLES

The following example adds a cell to the voltage_area shape:

```
prompt> set_voltage_area_shape -add_cells {uMid1} VOLTAGE_AREA_SHAPE_1
1
```

This example moves a voltage_area_shape one position up.

```
prompt> set_voltage_area_shape -raise VOLTAGE_AREA_SHAPE_1
1
```

This example moves a voltage_area_shape one position down.

```
prompt> set_voltage_area_shape -lower VOLTAGE_AREA_SHAPE_1
1
```

This example brings the voltage_area_shape to the top of the shape stack.

```
prompt> set_voltage_area_shape -top VOLTAGE_AREA_SHAPE_1
1
```

This example moves the VOLTAGE_AREA_SHAPE_2 below VOLTAGE_AREA_SHAPE_1

```
prompt> set_voltage_area_shape -below VOLTAGE_AREA_SHAPE_1 VOLTAGE_AREA_SHAPE_2
1
```

SEE ALSO

create_voltage_area(2)
create_voltage_area_shape(2)
get_voltage_area_shapes(2)
get_voltage_areas(2)
remove_voltage_area_shapes(2)
remove_voltage_areas(2)
report_voltage_areas(2)

set_vsdc

Generates a setup information file in VSDC format for efficient compare point matching in formal verification tools.

SYNTAX

boolean **set_vsdc**
[file_name]
[-append]
[-replace]
[-off]

Data Types

file_name string

ARGUMENTS

file_name

Specifies the file name that the tool should use for recording VSDC setup information. If this command is not issued, the VSDC file is not created. The path can be absolute or relative to the tool's current working directory. If the file exists, an error will result unless *-append* or *-replace* is specified. You must specify the file name unless *-off* is specified.

-append

Append the setup information to an existing file instead of creating a new one. If another VSDC file is already open, it is closed before opening the specified file. If the specified file does not exist, this option is ignored.

-replace

Overwrite the specified file, if it exists, with the new one.

-off

Close the VSDC file and stop recording setup information. No other options can be combined with *-off*.

DESCRIPTION

This command causes the tool to start recording setup information for formal verification tools. (If you are using Formality, the Synopsys formal verification product, you should use the **set_svf** command, instead.) The VSDC file that is produced can be used by some verification tools during the matching step to facilitate the alignment of compare points.

The VSDC file is a series of Tcl commands stored in plain-text format.

The VSDC file is used to account for changes to the hierarchy or to netlist object names that occur during synthesis. Certain operations performed by the compile command perform transformations that are also recorded in the VSDC file.

Because the VSDC output is buffered internally, the file might not be complete until recording is stopped. You can stop recording by running **set_vsdc** *-off* or by exiting from the tool. Running **set_vsdc** with a new file name will write out and close the original VSDC file before starting the new one.

This command returns a Boolean value indicating whether **set_vsdc** succeeded (true) or not (false).

EXAMPLES

In this example, we query the name of the VSDC file and find out that it has not been opened yet. Then we specify the file name and repeat the query.

```
prompt> get_vsdc
VSDC guidance is not enabled.
1
prompt> set_vsdc cache_ctrl.vsdc
1
prompt> get_vsdc
The current VSDC file is /project/chip/cache_ctrl/cache_ctrl.vsdc.
1
```

SEE ALSO

get_vsdc(2)
set_svf(2)
Formality User Guide

set_vt_filler_rule

Specifies which filler cells should be inserted by the **create_vtcell_fillers** command depending on the cells on the left and right of the gap being filled.

SYNTAX

status **set_vt_filler_rule**
-filler_cells *lib_cells*
-vt_type {*vt_type1 vt_type2*}
[-quiet]

Data Types

lib_cells	collection
vt_type1	string
vt_type2	string

ARGUMENTS

-lib_cells *lib_cells*

Specifies the list of filler cells to be used between the specified vt_types. When the **create_vtcell_fillers** command is issued the tool adds the filler cells in the order that you specify. Filler cells must be specified from the largest to smallest.

-vt_type {*vt_type1 vt_type2*}

Specifies the vt_types to insert cells between. The cells specified with the **-filler_cells** option will be inserted when the **create_vtcell_fillers** command is called. The string default is a keyword and is used to designate cells that have no vt_type assigned.

-quiet

Reduces the information written out by the command. When you specify this option, the command prints minimal information about the current VT settings.

DESCRIPTION

This command specifies which filler cells should be inserted by the **create_vtcell_fillers** command depending on the cells on the left and right of the gap being filled.

EXAMPLES

The following example shows how specify inserted A type filler cells between gaps that are bordered by vtA type cells.

```
prompt> set_vt_filler_rule \
  -filler_cells {myLib/filler2x  myLib/filler1x} \
  -vt_type { default default }

prompt> set_vt_filler_rule \
  -filler_cells {myLib/fillerA2x  myLib/fillerA1x} \
  -vt_type { vtA vtA }
```

SEE ALSO

set_cell_vt_type(2)
create_vtcell_fillers(2)
create_stdcell_fillers(2)

set_working_design

Sets the current working design.

SYNTAX

status **set_working_design**
-push *cell*
| -pop
[-level *level*]

Data Types

cell collection of one cell
level integer >= 0

ARGUMENTS

-push *cell*

Specifies a cell whose referenced design is pushed onto the stack. You can specify the cell by using either a cell name or a collection that contains one cell object.

This option is mutually exclusive with the **-pop** option. You must use one of them.

-pop

Pops the current working design out of the stack.

This option is mutually exclusive with the **-push** option. You must use one of them.

-level *cell*

Level to pop design stack.

This option works only with the **-pop** option.

DESCRIPTION

This command sets the working design by using a push or pop operation.

The command returns a status value that indicates success or failure.

When you run the **set_working_design** command, the tool maintains a stack of working designs. You can use the

get_working_design_stack command to check this stack and use the **set_working_design_stack** command to switch between different stacks.

The **current_design** command can also set the working design stack. However, it clears the destination stack. For more information, see the **current_design** man page.

When pushing to a child design, this command implicitly opens the child design if it is not yet opened. If you specify a soft macro cell, the open mode is edit if library is opened in edit mode. Otherwise, the open mode is read.

After you finish editing the child design, you can run the **set_working_design -pop** command to pop the child design from the stack and return to its parent design. The changes to the child design are automatically reflected in its ancestor designs.

EXAMPLES

The following example sets the current working design to a child design, BLENDER_2, which is instantiated by the I_ORCA_TOP/I_BLENDER_4 soft macro cell. After performing some operations, the example pops the child design from the stack and returns to the parent design, ORCA.

```
prompt> set_working_design -push I_ORCA_TOP/I_BLENDER_4
1
prompt> current_design
{BLENDER_2}
(some operations performed ...)
prompt> set_working_design -pop
1
prompt> current_design
{ORCA}
```

SEE ALSO

close_blocks(2)
current_design(2)
get_working_design_stack(2)
set_working_design_stack(2)
save_block(2)

set_working_design_stack

Sets the current working design stack.

SYNTAX

collection **set_working_design_stack**
design

Data Types

design collection

ARGUMENTS

design

Specifies the design whose stack is set as the current design stack. You can specify the design by name, name pattern, or a collection. If you specify a collection, the collection must contain only one design.

This is a required argument.

DESCRIPTION

This command sets the current working design stack by specifying a top-level design. The command returns a collection that contains the current working design stack.

The current working design is set to the top of target stack, which is the latest design pushed onto that stack. If the stack contains more than one design, the current working design is not the top-level design.

See the **current_design** man page for information about how to switch to the target top-level design and clear the stack.

EXAMPLES

The following example shows that the **set_working_design_stack** command sets the working design stack without clearing it.

```
prompt> current_design
{TOP}
prompt> set_working_design_stack ORCA
```

```
{ORCA BLENDER_2}
prompt> current_design
{BLENDER_2}
```

SEE ALSO

set_working_design(2)
get_working_design_stack(2)
current_design(2)

set_wrapper_configuration

Sets the default wrapper configuration for the design.

SYNTAX

status **set_wrapper_configuration**
[-reuse_threshold **threshold_value**]
[-chain_count **integer**]
[-test_mode *mode_name*]
[-hier_wrapping enable | disable]
[-mix_cells true | false]
[-mix_with_scan_cells true | false]
[-add_wrapper_cells_to_power_domains enable | disable]
[-use_system_clock_for_dedicated_wrp_cells enable | disable]
[-depth_threshold *threshold_value*]
[-gate_cells none | existing_cg | all]
[-feedthrough_chains disable | enable]
[-input_wrapper_cells *cell_list*]
[-output_wrapper_cells *cell_list*]
[-hold_mux_for_shared_wrapper_cells enable | disable]
[-safe_state 0 | 1 | none]

Data Types

threshold_value integer
mode_name string
cell_list string

ARGUMENTS

-reuse_threshold *threshold_value*

Specifies the maximum number of I/O registers that can be reused as shared wrapper cells.

When the number of I/O registers detected for a port exceeds the specified threshold value, a dedicated wrapper cell is added to the port.

The default value is 0.

If the option is set to 0, tool will use the maximized reuse flow and all I/O registers associated with a port are reused as shared wrapper cells.

If the option is set to -1, tool will use dedicated wrapper flow for all the ports in the design.

-chain_count *chain_count*

Specifies the maximum number of wrp_of chains.

When you do not use the **-chain_count** option, the number of wrp_of chains are determined by the lowest value of either -inputs or -outputs option of set_scan_compression_configuration command.

-test_mode ***mode_name***

Specifies the test mode to which the specification applies.

Only the following options can be specified with this option when the value of the option is not **all**:

-chain_count
-mix_cells

The values of all other options are taken from the wrapper configuration specification for the **all** mode.

The default is **all** (the specification applies to all modes).

-hier_wrapping enable | **disable**

Enables or disables hierarchical wrapping.

Hierarchical wrapping is supported only when there is only one scan mode synthesized in DFT insertion.

When hierarchical wrapping is enabled, the following core wrapping functionality is used:

All input wrapper cells are stitched into one or more input wrapper chains, controlled by an input shift-enable signal.

All output wrapper cells are stitched into one or more output wrapper chains, controlled by an output shift-enable signal.

Separate input and output control signals are used for the following wrapper control signal types:

- Wrapper shift signal
- Wrapper capture signal (if the input or output wrapper chains include a dedicated wrapper cell)
- SAFE mode enable signal (if the input or output wrapper chains include a wrapper cell that drives a SAFE value)

There is no wrapper mode logic added to the design.

The default is **disable**.

-mix_cells true | **false**

Specifies if wrapper cells for input or output ports can be mixed in a wrapper chain.

If this option is set to **false**, all wrapper cells in a wrapper chain are either associated with input ports or output ports but not both.

If this option is set to **true**, input and output wrapper cells can be mixed in the same wrapper chain.

The default is **false**.

-mix_with_scan_cells true | **false**

Controls whether scan cells can be mixed with wrapper cells on the same scan chain.

-add_wrapper_cells_to_power_domains enable | **disable**

Enables or disables the addition of dedicated wrapper cells to associated power domain hierarchies.

This option has no effect if there are no power domains in the design or no dedicated wrapper cells are added during wrapper insertion.

When this option is enabled, the tool identifies the hierarchy in which a dedicated wrapper cell is added by using the following

method.

If the port of the dedicated wrapper cell is connected to an I/O register, the top-level hierarchy of the power domain of the I/O register is used as the location for the dedicated wrapper cell.

If no I/O register is found for the port of the dedicated wrapper cell, the tool checks if the port is connected to a CTL modeled instance. If so, the top-level hierarchy of the power domain of the CTL modeled instance is used as the location for the dedicated wrapper cell.

If neither I/O register nor CTL modeled instance is found to be connected to the port of the dedicated wrapper cell, the top-level hierarchy of the first connected gate is used as the location for the dedicated wrapper cell.

If the I/O register or CTL modeled instance or the gate connected to the port of the dedicated wrapper cell does not have a power domain, the dedicated wrapper cell is added to the top-level hierarchy.

When this option is enabled, no additional hierarchy is created for dedicated wrapper cells added to the design.

When this option is enabled, any user-specified wrapper logic location (**set_dft_location ...**) is ignored.

The default is **disable** so that dedicated wrapper cells are added to the user-specified wrapper logic location if specified (**set_dft_location ...**) or to the top-level hierarchy.

-use_system_clock_for_dedicated_wrp_cells enable | disable

Controls whether the tool uses the system clock of I/O registers of the port as the clock for the dedicated wrapper cell.

When a dedicated wrapper cell is added to a port and this option is enabled, the tool uses the system clock of I/O registers of the port as the clock for the dedicated wrapper cell.

If the port is connected to a single I/O register and the tool adds a dedicated wrapper cell to the port, the clock of the associated I/O register is used as the clock for the dedicated wrapper cell.

The default wrapper clock is used for a dedicated wrapper cell if the associated port is not connected to any I/O registers.

The tool issue an error if more than one I/O register is found for a port and all of these I/O registers do not use the same clock domain and you did not specify any clock for the port. The command is terminated in such a case.

If you specify a clock for such a port, the tool treats the error as a warning and uses the specified clock as the clock for the dedicated wrapper cell.

The default is **enable**.

-depth_threshold *threshold_value*

Specifies the maximum combinational depth of I/O registers that can be reused as shared wrapper cells for a port.

The combinational logic depth is defined as the maximum number of combinational logic levels, including buffers and inverters, between a port and an associated I/O register. For a given port, if the combinational logic depth of any associated I/O register exceeds this value, a dedicated wrapper cell is added to the port.

There is no default for this option. When not specified, the check is not performed and all I/O registers associated with a port are reused as shared wrapper cells, subjective to other checks.

-gate_cells none | existing_cg | all

Controls whether clock-gating logic is inserted or modified to reduce power consumption in core wrapping modes.

This feature implements the following behavior for integrated clock-gating cells:

- In EXTEST (outward-facing) modes, clocks to core register cells are disabled.
- In SAFE mode, clocks to both core register cells and wrapper cells are disabled.

When this option is set to **existing_cg**, the tool modifies only existing integrated clock-gating cells. Both the functional and test enable pin connections are modified.

When this option is set to **all**, the tool modifies existing integrated clock-gating cells, and it also inserts integrated clock-gating cells for more aggressive power reduction. (You must set the **dft.test_icg_p_ref_for_dft** application option to specify the rising-edge ICG library cell to insert.)

The default is **none**, which disables the feature.

-feedthrough_chains enable | **disable**

Specifies whether feedthrough chains are created in transparent modes.

When this option is set to **enable, feedthrough chains with dummy registers are created in inward-facing test modes. No extra test mode is required or created to enable the support. This ensures that the core can be integrated in top-level designs that have a top-level codec.**

The default is **disable**, such that feedthrough chains are not created in inward-facing test modes. In this case, you no longer need to use dedicated wrapper-mode scan I/Os for the core's outward-facing modes. The resulting core contains no feedthrough chains, and it can be integrated in top-level designs that do not have a top-level codec.

-input_wrapper_cells ***cell_list***

Specifies additional scan cells to be classified as output-related registers to include in the input wrapper chain. The input is accepted as a simple list. Wildcards and collections are not supported.

This option is not cumulative; new specifications overwrite previous specifications. Specify an empty list to remove a previous specification.

-output_wrapper_cells ***cell_list***

Specifies additional scan cells to be classified as input-related registers to include in the output wrapper chain. The input is accepted as a simple list. Wildcards and collections are not supported.

This option is not cumulative; new specifications overwrite previous specifications. Specify an empty list to remove a previous specification.

-hold_mux_for_shared_wrapper_cells enable | **disable**

Enables or disables a hold MUX for shared wrapper cells.

When a hold MUX is added, shared wrapper cells hold the state when the cell is not in shift mode in test operation.

The default is not to add a hold MUX to shared wrapper cells.

-safe_state 0 | **1** | **none**

Specifies the safe state value to be used for wrapper cells added by core wrapping.

The default is **none**.

DESCRIPTION

This command sets the default wrapper configuration for the design.

EXAMPLES

```
set_wrapper_configuration -reuse_threshold 4 -chain_count 6
```

SEE ALSO

set_scan_compression_configuration(2)
preview_dft(2)

setenv

Sets the value of a system environment variable.

SYNTAX

string **setenv** *variable_name new_value*

string *variable_name*

string *new_value*

ARGUMENTS

variable_name

Names of the system environment variable to set.

new_value

Specifies the new value for the system environment variable.

DESCRIPTION

The **setenv** command sets the specified system environment *variable_name* to the *new_value* within the application. If the variable is not defined in the environment, the environment variable is created. The **setenv** command returns the new value of *variable_name*. To develop scripts that interact with the invoking shell, use **getenv** and **setenv**.

Environment variables are stored in the Tcl array variable **env**. The environment commands **getenv**, **setenv**, and **printenv** are convenience functions to interact with this array.

The **setenv** command sets the value of a variable only within the process of your current application. Child processes initiated from the application using the **exec** command after a usage of **setenv** inherit the new variable value. However, these new values are not exported to the parent process. Further, if you set an environment variable using the appropriate system command in a shell you invoke using the **exec** command, that value is not reflected in the current application.

EXAMPLES

The following example changes the default printer.

```
shell> getenv PRINTER
laser1
shell> setenv PRINTER "laser3"
laser3
shell> getenv PRINTER
laser3
```

SEE ALSO

exec(2)
getenv(2)
unsettenv(2)
printenv(2)
printvar(2)
set(2)
sh(2)
unset(2)

setup_performance_via_ladder

Setup for automatic performance via ladder flow. Generate via rules based on the analysis of layers and associate via rules on lib_pins.

SYNTAX

status **setup_performance_via_ladder**
-max_layer layer_name
-lib_pins lib_pin_names
-association_file association_file_name
-rule_file rule_file_name
-effort low|medium|high|ultra
-stack_via none|alone|composite
-reset
-ignore_dont_use
-stagger
-smaller

Data Types

layer_name string
lib_pin_names string or collection of lib_pins
association_file_name string
rule_file_name string

ARGUMENTS

-max_layes *layer_name*

Specifies max_layer of via ladders. This should not be higher layer than the max_layer in the routing rule.

-lip_pins *lib_pin_names*

Specifies lib_pin_names which need performance via ladder candidates.

-association_file *association_file_name*

Specifies output via ladder association file name. (Default: auto_perf_via_ladder_association.tcl)

-rule_file *rule_file_name*

Specifies output via rule file name. (Default: auto_perf_via_ladder_rule.tcl)

-effort *low*|*medium*|*high*|*ultra*

Specifies effort level. Higher effort will generate more via ladder associations to more lib pins. (Default: low)

-stack_via *none*|*alone*|*composite*

Generate stack via rules and association files. (Default: none)

-reset

Reset existing via ladder candidates from lib_pins.

-ignore_dont_use

Allow via_ladder candidates on lib_cells with dont_use attribute.

-stagger

Generate stagger table.

-smaller

Associate smaller via ladders first.

DESCRIPTION

This command sets automatic performance via ladder flow. It generates via rules based on the analysis of layers and associates via rules on each lib_pin. This command returns 1 if succeeded, 0 otherwise.

When -lip_pins option is not used, the tool searches for all lib cells to select the lip pins to associate. It associates max 2 via ladder candidates per pin, by default. You can increase the number of the candidates with -effort option. With the increased effort level, more candidates are associated to more lib cells. With -effort ultra, almost every lib cells will have the candidates.

When -lip_pins option is used, all those lib pins will have the candidates. The effort level will decide the number of candidates per pin.

If a lib cell already have via ladder candidates before running this command, those candidates will be preserved in the association file. If you want to remove existing candidates, use -reset option.

This command associates bigger via ladder candidates first, by default, out of many possible candidates. This may hurt via ladder insert success ratio in congested design where smaller via ladder may be desirable. With -smaller option, the tool associates smaller via ladder candidates first.

This command can generate stack vias. When -stack_via alone option is used, the tool generates stack via rules and association files, where generic via ladder rules are not generated. -effort and -reset option will be ignored in this mode. -stack_via composite will generate stack via rules and association files using both via ladders and stack vias.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the options.

EXAMPLES

The following is the usage examples.

```
prompt> setup_performance_via_ladder
1

prompt> setup_performance_via_ladder -effort ultra -reset
1

prompt> setup_performance_via_ladder -effort high -smaller
1

prompt> setup_performance_via_ladder -reset -stack_via alone
1

prompt> setup_performance_via_ladder -reset -stack_via composite -effort ultra
1

prompt> setup_performance_via_ladder -max_layer M10 -lib_pins [get_lib_pins */*D12*/Z]
1
```

SEE ALSO

create_via_rule(2)
remove_via_rules(2)
report_via_rules(2)
set_via_ladder_candidate(2)
reset_via_ladder_candidates(2)
report_via_ladder_candidates(2)

sh

Executes a command in a child process.

SYNTAX

string **sh** [*args*]

string *args*

ARGUMENTS

args

Command and arguments that you want to execute in the child process.

DESCRIPTION

This is very similar to the **exec** command. However, file name expansion is performed on the arguments. Remember that quoting and grouping is in terms of Tcl. Arguments which contain spaces will need to be grouped with double quotes or curly braces. Tcl special characters which are being passed to system commands will need to be quoted and/or escaped for Tcl. See the examples below.

EXAMPLES

This example shows how you can remove files with a wildcard.

```
prompt> ls aaa*
aaa1    aaa2    aaa3
prompt> sh rm aaa*
prompt> ls aaa*
Error: aaa*: No such file or directory
    Use error_info for more info. (CMD-013)
```

This example shows how to grep some files for a regular expression which contains spaces and Tcl special characters:

```
prompt> exec cat test3.out
blah blah blah
```

```
blah blah blah c blah
input [1:0] A;
output [2:0] B;
prompt>
prompt> sh egrep -v {[ ]+c[ ]+} test3.out
blah blah blah
prompt>
prompt> sh egrep {t[ ]+\[} test3.out
input [1:0] A;
output [2:0] B;
```

SEE ALSO

exec(2)

shape_blocks

Shapes and places the shaping objects (physical blocks, power domains or voltage areas, and move bounds) with respect to constraints and utilization requirements. If no utilization constraint is set on the shaping object, the design's utilization will be used. The command uses macro packing to place the top-level macros. If the block grid is created for the block before shaping, it will also align the block shape to the specified block grid. If the placement abstracts of blocks have not been created before this command, shape_blocks will automatically create them.

SYNTAX

int **shape_blocks**
[-channels true | false]
[-constraint_file *file_name*]
[-incremental congestion_driven | target_utilization_driven]
[-pg_strategy *pg_strategy_name*]
[-host_options *host_options_name*]

Data Types

file_name string
pg_strategy_name string
host_options_name string

ARGUMENTS

-channels true | false

Specifies whether channels are created between blocks. Channels can be inserted between blocks or between blocks and the core boundary. By default, the option is true and the command inserts channels.

If channels are required, the channel size is automatically calculated based on the required routing resources among hierarchies and the allowed utilization of each physical hierarchy.

You can also specify the required minimal channel size with the **set_shaping_options -min_channel_size** command. The larger value of user-specified channel size and automatically calculated channel size is used to create channels. When **-min_channel_size** is specified, the automatically calculated channel size is based only on the required routing resources, not on the allowed utilization.

-constraint_file *file_name*

Specifies the file that contains user constraints to control the channel sizes, aspect ratio, utilization, fixed shapes, and relative placement. The supported constraint formats are as follows.

- *Constraint types*

The following keywords define shaping constraints:

- block related keywords
- *allowed_orientation*
- *aspect_ratio*
- *utilization*
- *boundary*
- edge related keywords
- *boundary_channel_size*
- *channel_size*
- *guard_band*
- grouping related keywords
- *define_group*
- relative placement related keywords
- *arrange_in_box*
- *arrange_in_array*

Constraints can take parameters, which is illustrated in the following example of defining channels around a block CPU instantiated within block B:

```
block B {
  channel_size {
    // This constraint only applies to sub-block CPU.
    block_inst CPU;

      // Default: any channel surrounding CPU should be 20u wide.
      // Except left and right size of CPU // which are between 30u
      and 50u.  min = 20, max = 20; left, right: min = 30, max = 50;
      } // Other constraints within the scope of block B go here }
```

- *Scoping and grouping rules*

Global constraints applied to any block are defined at the global scope. These constraints do not have an encompassing block scope. The only constraints that can be defined in global scope are: boundary_channel_size, channel_size, guard_band and utilization. For example

```
  utilization { target: 0.3; }
```

specifies a target utilization of 30% for any shapeable region in this design, unless it is overridden by any later constraint.

guard_band and channel_size can also be specified in the global scope. For example

```
  guard_band { width: 2; height: 3; }
  channel_size { size = 5; }
  block TOP {
    // define constraints in block scope
    ......
  }
```

specifies a guard_band for any voltage area of 2u in width and 3u in height. It also specifies the channel size of 5u between any blocks/VAs.

The default guard band size is zero. For non-abutted floorplan (i.e., option **-channels true** is applied), the default channel size is calculated to minimize the channel routing congestion.

The space between two neighboring blocks is sum of the outer keepouts and channel size for the blocks. The space between two neighboring voltage areas is sum of guard band sizes and channel size.

The constraints defined in the global scope can be overwritten by the constraints defined in the block scope, as shown in the following

example.

Constraints applicable to objects within a particular block are defined inside a block scope:

```
block <reference_name> {
   constraint { }
   constraint { }
   ......
}
```

The reference name identifies the block being constrained. The constraints defined within the block scope can only refer to objects visible in the physical block, or to the physical block itself using the keyword current_block. The constraints cannot refer to objects of another physical block.

Constraints specific to a voltage area in a given block are specified using a voltage area scope. For example

```
guard_band { width: 1; height: 1; }
block BLK1 {
   voltage_area VAS1 {
      contents: voltage_area VAS2, block_inst BLK2;

      // VAS1 and nested voltage areas, like VAS2 get 40u guard band
      // even though the global guard band is 1u. This constraint is
      // limited to voltage areas of this block, BLK1.
      guard_band { width: 40; height: 40; }

      ...
   }
   guard_band {
     voltage_area VA1;
     width: 20; height: 20;
   }
}
```

Specifies the following:

- The voltage area VAS2 and block BLK2 are to be shaped inside VAS1
- A guard band of 40u for VAS1 and any voltage area nested inside (but within the current block)
- A guard band of 20u for voltage area VA1
- A guard band of 1u for other voltage areas

You can define new groups of objects like block instances, voltage areas and other groups with the define_group constraint.

```
block BLK1 {
   define_group GRP1 {
      // E, F, MB1, and VA3 make up group GRP1
      contents: block_inst E, block_inst F,
            move_bound MB1,
            voltage_area VA3;

   }

   arrange_in_array {
      // Refers to the group defined above
      contents: group GRP1,
            block_inst J;
      direction: east;
   }
```

```
}
```

Groups cannot have partial overlap, but nesting is allowed. For example, the following constraint specifies that two blocks, E and F, must be arranged from top to bottom, and shaped along VA3:

```
block BLK1 {
   define_group GRP1 {
      contents: block_inst E, block_inst F,
            voltage_area VA3;

      define_group GRP2 {
         contents: block_inst E, block_inst F;
         arrange_in_array {
            contents: block_inst E, block_inst F;
            direction: south;      // from top to bottom
         }
      }
   }
   // GRP1 shadows E, F and VA3. Constraints at this point
   // can refer to GRP1, (not to E, F and VA3) Those have
   // to be defined inside GRP1 or GRP2.
}
```

In the scope where a group is defined, the group will shadow the objects it contains to prevent inadvertent group overlap. In the example above, GRP1 will shadow E, F and VA3.

- *Constraints*

- Boundary channel constraint

The boundary_channel_size constraint reserves a space between a block or voltage area and its nested regions. The boundary_channel_size and channel_size constraints applying to the same channel are combined by taking the maximum of the two. When a boundary_channel_size constraint is defined inside the global scope, it applies to the boundary inside any block or voltage area. Alternatively, the boundary_channel_size is defined within a block scope, and is applied to the current_block or to a voltage area, specifying the boundary channel's size along the inside of that region's boundary. For example,

```
block CPU {
   boundary_channel_size {
     current_block;
     left: min = 3, max = 5;
     right, bottom: min = 5, max = 99;
   }
}
```

specifies that any voltage area or block instance shaped within CPU is to stay at least 3u to 5u from the left side of the boundary, and 5u to 99u from the right and bottom side of the boundary.

- Utilization constraint

Utilization constraints specify a minimum, target or maximum cell utilization of a shapeable region. For example, requesting a 53% target utilization and setting an upper limit of 70% and a lower limit of 40% is achieved by:

```
utilization {
   target: 0.53;
   max: 0.7;
   min: 0.4;
}
```

Utilization of block instances cannot be specified individually, because multiple instantiations (MIB's) must follow the same constraint.

Instead, you define the constraint for the reference design, say X:

```
block X {
   utilization { target: 0.4; }
}
```

A utilization constraint that does not explicitly name any objects will be applied to the shapeable regions in the current scope (limited to the current block). In the next example a 60% utilization target is set for VA1, VA2 and VA3 (as both are part of G2), but not to block_inst B.

```
define_group G2 { contents: voltage_area VA2, voltage_area VA3; }
define_group G1 {
  contents: voltage_area VA1, group G2, block_inst B;
  utilization { target: 0.6; }
}
```

The above is equivalent to:

```
define_group G2 { contents: voltage_area VA2, voltage_area VA3; }
define_group G1 { contents: voltage_area VA1, group G2, block_inst B; }
utilization { group G1; target: 0.6; }
```

- Aspect ratio constraint

The target aspect ratio of a region can be specified for blocks that are placed with an arrange_in_box constraint:

```
  block RESET {
    aspect_ratio {
       current_block;
       target: width = 3, height = 4;
    }
  }
```

- Guard band constraint

Guard band constraints only affect shapeable voltage areas, and control the width and height of their guard bands. If you specify a voltage area, the constraint will apply to that voltage area only (not recursively). If you specify a group, the constraint will apply to the voltage areas of that group (not recursively). In the example below, a guard band of 20 is assigned to VA3 and VA6.

```
  block CPU1 {
    define_group GRP1 { contents: voltage_area VA6, block_inst L; }
    guard_band { voltage_area VA3, group GRP1; width: 20; height: 20; }
  }
```

If you leave out a list of regions, the constraint will be applied recursively to any voltage area within the encompassing scope and current block. If the encompassing scope is a voltage area, it is included as well. The example below, shows how a guard band of 10u is assigned to all the voltage areas within GRP2 and below (within the same block): VA1, VA2, VA4 and VA5. It also shows how nesting between voltage areas and between move bounds.

```
  block CPU1 {
    // define nesting between voltage areas
    voltage_area VA1 {
      contents: voltage_area VA4, voltage_area VA5;
    }
    // define nesting between move bounds
    move_bound mb1 {
     contents: move_bound mb2;
    }
    define GRP2 {
```

```
    contents: voltage_area VA1, voltage_area VA2, block_inst K;
    guard_band { width: 10; height: 10; }
  }
}
```

- Channel size constraint

A channel_size constraint specifies the size of a channel between regions.

```
channel_size {
  // Primary objects.
  block_inst CPU1, block_inst CPU2;

  // Secondary (neighboring) objects.
  neighbors: current_block;

  // Any channel between block (CPU1 or CPU2) and current_block is 10.
  // For the channel along the left, right side of CPU1 (or CPU2) and
  // neighboring the boundary of current_block,
  // limit the size of the channel to [3, 5].
  size=10;
  left, right: min=3, max=5;
}
```

A channel_size constraint contains an object specifier followed by one or more range specifiers. Object specifiers can be:

- empty, the channel size constraint applies to any object in the current scope.
- a single list, specifying primary objects:
- block_inst followed by an identifier
- voltage_area followed by an identifier
- move_bound followed by an identifier
- group followed by an identifier
- current_block identifying the boundary of the encompassing block
- between a primary object and its neighbor: two lists specified similarly as above. Applies to all possible pairs formed by an object from the primary list and an object from the secondary list.

A range specifier is an edge specifier followed by a limit specifier. They can be:

- empty, in which case the channel size constraint applies to any edge.
- side specific: any of left, right, bottom, top to identify the sides of the primary object.

A limit specifier defines the size range of the constrained channel. It can be one of the following three:

- min = <IDENTIFIER> : specifies a minimum size, maximum unconstrained.
- max = <IDENTIFIER> : specifies a maximum size, minimum is bounded by default (0).
- min = <IDENTIFIER>, max = <IDENTIFIER> : limits the minimum and maximum size of this channel.
- size = <IDENTIFIER> : specifies an exact size.

The channel_size constraint specified last overwrites any previously defined channel_size constraints with equal object specifier. A more specific constraint takes precedence over less specific constraints. Channel size constraints defined inside a block scope are more specialized than channel size constraints defined at the global scope. Specialization is next determined by the object specifier, defined to have increasing order of precedence as follows: empty, object list, object list with neighbor list.

- Relative placement constraint

The arrange_in_array and arrange_in_box constraints instruct the shaper to generate a specific region layout.

The arrange_in_array constraint has the contents list and a direction. The shaper places the regions in the contents list in an array following the specified order, growing the array in the specified direction. Valid directions are: north, west, south, east.

```
block B {
  arrange_in_array {
    contents: block_inst A, voltage_area VA1, group G2;
    direction: west;
  }
}
```

In above example, group A, VA1, and G2 are placed right-to-left in a horizontal array. Change the order to left-to-right by setting the direction to east.

The arrange_in_box constraint places the contents at the specific location. Optionally, the arrange_in_box constraint carries an alignment point (default at center). When an alignment point is specified, the shaper tries to align that point with the target location.

```
voltage_area VA1 {
   arrange_in_box {
      block_inst A;
      // Optional alignment point, default is center.
      alignment_point: x = 1, y = 0;
      location: x = 0.5, y = 0.5;
   }
}
```

In above example, block instance A is placed within VA1's shape. A's lower right corner (1, 0) is to be aligned with VA1's center.

In the following example, the shaper first shapes block C (if it does not already have a given shape) and then applies the arrange_in_box constraints in their order of specification. Finally, the arrange_in_array constraints are applied. In the example above, the shaper places A, GRP1 (= D, E, F, G, VA2). This leaves H and X to be shaped into the cavity defined by C minus the area assigned to A and GRP1. The shaper then arranges H on the left of X. The region layout within group GRP1 is also predetermined. A subgroup GRP2 consisting of block instances D and E is defined, and given a center location. The coordinates define a location within the encompassing region's bounding box, here GRP1, and range from (0, 0), the lower left corner, to (1,1), the upper right corner. Again, the arrange_in_box constraint takes precedence over the arrange_in_array constraint. Note that VA2 is left unconstrained. Finally, within group GRP2, the block instances D and E are arranged from top to bottom. This constraint is applied once GRP2 has been assigned a shape.

```
// Block C contains the following regions for shaping: [A, D, E,
// F, G, H, X, VA2]. Blocks D, E, F, G and VA2 should be kept together,
// other blocks can be placed without restriction
block C {

   define_group GRP1 {
   // The following regions make up group GRP1
   // They must be within C's scope.

      contents: block_inst D, block_inst E, block_inst F,
            block_inst G, voltage_area VA2;

      define_group GRP2 {
         arrange_in_array {
           contents:  block_inst D, block_inst E;
           direction: south;
         }
```

```
        }

        // Order is important. First apply the arrange_in_box,
        // then apply the arrange_in_array constraint
        arrange_in_box {
           // Referring to D and E is not allowed here, because
           // they are grouped into GRP2
           group GRP2;
           location: x = 0.5, y = 0.5; // place in the middle
        }

        // Note: this implicitly defines an unnamed group of F and G.
        arrange_in_array {
           contents: block_inst F, block_inst G;
           direction: east;
        }
     }

     arrange_in_box { block_inst A; location: x = 1, y = 0.5; }
     arrange_in_box { group GRP1; location: x = 1, y = 0.5; }
     arrange_in_array {
        contents: block_inst H, block_inst X;
        direction: east;
     }
  }
```

- Boundary constraint

To consider the current boundary of a region as rigid, define the boundary as rigid within that scope, and provide a shape. Rigid regions will still be moved to a suitable location, but their shapes will not be changed.

For example,

```
  voltage_area VA1 {
     // Only move boundary of VA1, do not change shape
     boundary {
        type: rigid;
        shape: {{0 0} {300 600}};
        // or rectilinear shape as follows
        // shape: {{0 0} {{600 0} {600 1100}
        // {300 1100} {300 500} {0 500} {0 0}};
     }
  }
```

- Allowed orientation constraint

With this constraint, the orientation of individual MIB instances can be restricted to a set of allowed orientations. The specified orientations can be R0, R180, MX and MY.

The specified values are expected to be a subset of the allowed orientations of the reference designs of the given block instances. If a reference design has an associated block grid, the allowed orientations follow from that grid. Otherwise, they follow from the design's allowable_orientations attribute.

If the specified set of orientations is not a subset, it will be ignored silently.

Here's an example restricting the orientation of MIB instances B1, B2, B3 and B4.

```
  block Top {
     // Only move boundary of VA1, do not change shape
```

```
    allowed_orientation {
      block_inst B1, block_inst B2;  // MIB block instances being constrained.
      { R0 R180 };                   // Allowed orientations are R0 and R180.
    }
    allowed_orientation {
      block_inst B3, block_inst B4;
      { MY };
    }
  }
```

-incremental congestion_driven | target_utilization_driven

Specifies that congestion-driven or target utilization-driven incremental shaping is applied. By default, neither shaping strategy is used.

If you specify **-incremental congestion_driven**, routing-based congestion information must be available for the incremental shaping to get the initial channel congestion. You must run global routing before using this option. For the congestion driven incremental shaping, all MIBs will not be moved. Only channel congestion between blocks are considered in the incremental shaping.

If you specify **-incremental target_utilization_driven**, the tool incrementally changes the block shapes to achieve the target utilization specified for each block. If no target utilization is specified, the command uses the design utilization as the target for all blocks with respect to the utilization slack specified by **set_shaping_options -utilization_slack**. For the target area driven incremental shaping, option **-channel false** is not supported.

This option and option **-constraint_file** are mutually exclusive. Only one of them can be specified.

-pg_strategy *pg_strategy_name*

Specifies the PG (power-ground) strategy that the **shape_blocks** command should respect so that the shape boundaries are not collinear with PG meshes.

If a mesh pattern, for example my_pattern, is created using **create_pg_mesh_pattern**, the PG strategy named smesh can be created with the following command:

set_pg_strategy smesh -pattern {{name: my_pattern}{nets: VDD VSS}} -core

When you run the **shape_blocks** command as follows and specify the smesh PG strategy:

shape_blocks -pg_strategy {smesh}

the tool creates shapes which are not collinear with the specified PG meshes. You must specify **-core** with the **set_pg_strategy** command for shaping to create the same PG meshes for the whole design.

-host_options

Specifies that the given host option should be used for distributed processing. Note that only creating the appropriate block views (if needed) will use distributed processing in this command.

DESCRIPTION

This command shapes and places physical blocks, power domains or voltage areas, and move bounds with respect to constraints and utilization requirements. The command also applies optimization to minimize feedthrough counts and interface wirelengths. If there is a power domain which has no corresponding voltage area, this command will create its associated voltage area. If the block grid is created for the block before shaping, it will also align the block shape to the specified block grid.

The **set_editability** command can be used to allow or prevent shaping on specific blocks. Use the **set_shaping_options** command to specify additional constraints for shaping, such as minimum channel size, channel blockages, and so on.

The tool automatically infers the nesting of move bound inside voltage area based on the contents such that voltage area gets the higher priority. If the elements of voltage area are top/hier1 and the elements of move bound are top/hier1/hier2, the move bound will be nested inside the voltage area based on the hierarchy tree such that all elements of the voltage area are in the same voltage area region. For this reason, it is not supported that the same element is used to define voltage area and move bound, i.e., both use top/hier1. In case that the contents are really the same, use top/hier1/* to define the elements of move bound.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following example creates an abutted floorplan.

```
prompt> shape_blocks -channels false
```

The following example creates a floorplan with channels and also respects the constraints specified in the **user_constraints** file.

```
prompt> shape_blocks -constraint_file user_constraints
```

Example of making manual adjustments to boundaries after automatic block shaping.

```
set_attribute [get_cells BLK1] -name boundary -value {{0 0} {1000 1000}}
set_attribute [get_voltage_area_shapes -of_objects VA2] -name boundary -value {{0 0} {1000 1000}}
set_attribute [get_bound_shapes -of_objects MB3] -name boundary -value {{0 0} {1000 1000}}
```

SEE ALSO

create_grid(2)
get_bound_shapes(2)
get_voltage_area_shapes(2)
set_editability(2)
set_shaping_options(2)

shell_is_in_cv_mode

Determines if the shell was invoked in concurrent verification mode.

SYNTAX

status **shell_is_in_cv_mode**

ARGUMENTS

The **shell_is_in_cv_mode** command has no arguments.

DESCRIPTION

This command is used to check if the shell is in concurrent verification mode; the shell was launched from cv_shell.

The command returns 1 if the shell is in concurrent verification mode, or a 0 otherwise.

EXAMPLES

The following example shows how to check the mode in which the shell has been invoked:

```
prompt> if { [shell_is_in_cv_mode] } { \
          echo "Running cv_shell" \
          enable_concurrent_verification -run_dir formality
        } else { \
          echo "Not in cv_shell mode" \
        }
```

signoff_calculate_hier_antenna_property

Calculates the hierarchical antenna properties of a hard macro.

SYNTAX

status **signoff_calculate_hier_antenna_property**
-diffusion_layers {*list_of_layers*}
[-top_cell_pin_only true | false]
[-treat_source_drain_as_diodes true | false]
[-report_diodes true | false]
[-poly_layers {*list_of_layers*}]
[-contact_layers {*list_of_layers*} |
[-v0_layers_between_m1_m0 {*list_of_layers*}]
[-m0_layers_for_poly_connection {*list_of_layers*}]
[-m0_layers_for_diffusion_connection {*list_of_layers*}]
[-contact_layers_between_m0_diffusion {*list_of_layers*}]]
[-gate_class1_marking_layers {*list_of_layers*}]
[-gate_class2_marking_layers {*list_of_layers*}]
[-gate_class3_marking_layers {*list_of_layers*}]

Data Types

list_of_layers list

ARGUMENTS

-diffusion_layers {*list_of_layers*}

Specifies the diffusion layers. Specify the layers by using the layer names or layer numbers from the technology file.

This option is required.

-top_cell_pin_only true | false

Controls how hierarchical nets are handled during antenna property extraction.

By default (true), only properties for nets that are connected to pins in the top level of the hard macro are considered.

If this option is set to false, properties for nets that are connected to pins in the child cells contained in the hard macro are also considered.

-treat_source_drain_as_diodes true | false

Controls how the tool treats the MOS source and drain.

By default (true), the MOS source and drain regions are treated as protection diodes and the MOS source and drain regions that

are connected to a net to be a part of the diode protection area available to that net.

If this option is set to false, they are not treated as protection diodes.

-report_diodes true | false

Controls whether or not the tool reports the location of generated protection diodes.

By default (true), the locations will be reported.

If this option is set to false, the locations will not be reported.

-poly_layers {*list_of_layers*}

Specifies the polysilicon layers. Specify the layers by using the layer names or layer numbers from the technology file.

-contact_layers {*list_of_layers*}

Specifies the contact layers (contact layers are the layers that connect polysilicon to metal1). Specify the layers by using the layer names or layer numbers from the technology file.

-v0_layers_between_m1_m0 {*list_of_layers*}

Specifies the v0 layers (v0 layers are the layers that connect metal1 to metal0). Specify the layers by using the layer names or layer numbers from the technology file.

The **-v0_layers_between_m1_m0** and **-contact_layers** options are mutually exclusive; specify only one.

-m0_layers_for_poly_connection *list_of_layers*

Specifies the m0 layers for poly connection. Specify the layers by using the layer names or layer numbers from the technology file.

The **-m0_layers_for_poly_connection** and **-contact_layers** options are mutually exclusive; specify only one.

-m0_layers_for_diffusion_connection *list_of_layers*

Specifies the m0 layers for diffusion connection. Specify the layers by using the layer names or layer numbers from the technology file.

The **-m0_layers_for_diffusion_connection** and **-contact_layers** options are mutually exclusive; specify only one.

-contact_layers_between_m0_diffusion *list_of_layers*

Specifies the contact layers (contact layers are the layers that connect metal0 to diffusion). Specify the layers by using the layer names or layer numbers from the technology file.

The **-contact_layers_between_m0_diffusion** and **-contact_layers** options are mutually exclusive; specify only one.

-gate_class1_marking_layers {*list_of_layers*}

Specifies the marking layers for gate thickness class 1. Specify the layers by using the layer names or layer numbers from the technology file.

-gate_class2_marking_layers {*list_of_layers*}

Specifies the marking layers for gate thickness class 2. Specify the layers by using the layer names or layer numbers from the technology file.

-gate_class3_marking_layers {*list_of_layers*}

Specifies the marking layers for gate thickness class 3. Specify the layers by using the layer names or layer numbers from the technology file.

DESCRIPTION

This command invokes the IC Validator tool using an automatically generated runset to calculate the hierarchical antenna properties of the current design (hard macro) and stores the values in the database.

The hierarchical antenna properties include gate size, routing area, and diode protection for all the ports of the macros. These properties are used by the router to check antenna rules on nets connected to the macro pins.

Before you use this command, you must open the design library and DESIGN view. The hierarchical antenna properties are stored in the FRAME view of the design. Both the DESIGN and FRAME views of the cell must exist before you run this command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the command to extract the hierarchical antenna properties of a hard macro with gate class 1 and gate class 2 marking layers.

```
prompt> signoff_calculate_hier_antenna_property \
  -diffusion_layers { 2 5 } -poly_layers { 7 } \
  -contact_layers  { 14 }  \
  -gate_class1_marking_layers { 12:12 } \
  -gate_class2_marking_layers { 24:12 }
```

The following example runs the command to extract the hierarchical antenna properties of a hard macro with a v0 layer between m1 and m0, an m0 layer for poly connection, and an m0 layer for diffusion connection.

```
prompt> signoff_calculate_hier_antenna_property \
  -diffusion_layers { 6 } -poly_layers { 17 } \
  -v0_layers_between_m1_m0 { 159 } \
  -m0_layers_for_poly_connection { 28 } \
  -m0_layers_for_diffusion_connection { 84 }
```

The following example runs the command to extract the hierarchical antenna properties of a hard macro with a v0 layer between m1 and m0, an m0 layer for poly connection, an m0 layer for diffusion connection, and a contact layer between m0 and diffusion.

```
prompt> signoff_calculate_hier_antenna_property \
  -diffusion_layers { 2 } -poly_layers { 7 } \
  -v0_layers_between_m1_m0 { 35 } \
  -m0_layers_for_poly_connection { 176 } \
  -m0_layers_for_diffusion_connection { 14 } \
  -contact_layers_between_m0_diffusion { 101 }
```

SEE ALSO

derive_hier_antenna_property(2)

signoff_check_design

Command to do different checks such as metal fill shorts.

SYNTAX

status **signoff_check_design**
[-short_with_metal_fill *true* | *false*]
[-read_frame_view {*list of blocks*}]

Data Types

list of blocks string

ARGUMENTS

-short_with_metal_fill *true* | *false*

Specifies to check for the shorts with metal fill shapes.

-read_frame_view {*list of blocks*}

Specifies the list of blocks for which frames are to be read. By default, the design view of all macros are used if available then the frame view. Wildcards are acceptable.

DESCRIPTION

This command launches the IC Validator tool with a runset created on the fly to do basic checks. The currently supported one is to check shorts with metal fill.

The command creates runset on the fly, launches ICV with the runset and creates an error data for violations.

Before running this command, you must setup the environment settings for the IC Validator tool. For more details about the IC Validator tool, see the *IC Validator Reference Manual*.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs design rule checking with maximum number of errors per rule set to 6000.

```
prompt> set_app_options -name signoff.check_design.max_errors_per_rule -value 6000
prompt> signoff_check_design
```

The following example runs IC Validator under the working directory named **my_signoff_run**.

```
prompt> set_app_options -name signoff.check_design.run_dir -value "my_signoff_run"
prompt> signoff_check_design
```

The following example runs IC Validator with user-defined option named **-turbo** and user given runset.

```
prompt> set_app_options -name signoff.check_design.user_defined_options -value "-turbo"
prompt> set_app_options -name signoff.check_design.runset -value "short_finder.rs"
prompt> signoff_check_design
```

SEE ALSO

signoff_check_drc(2)
signoff_create_metal_fill(2)

signoff_check_drc

Performs design rule checking on the current block using a foundry runset.

SYNTAX

status **signoff_check_drc**
[-error_data *errdata_name*]
[-check_all_runset_layers true | false]
[-select_layers *collection_of_layers*]
[-select_rules *list_of_rule_names*]
[-unselect_rules *list_of_rule_names*]
[-coordinates *region_spec*]
[-excluded_coordinates *region_spec*]
[-auto_eco true | false]
[-pre_eco_design *block_name*]

Data Types

errdata_name	string
collection_of_layers	collection
list_of_rule_names	list
region_spec	list
block_name	string

ARGUMENTS

-error_data *errdata_name*

Specifies the name of the error data file.

If you do not specify this option, the error data file is named signoff_check_drc.err.

-check_all_runset_layers true | false

Controls the layers on which to perform design rule checking.

By default (**false**), the command checks the design rules only on the routing layers (metal and via layers).

If **true**, the command checks the design rules on all layers defined in the runset, such as routing layers and device layers. When you set this option to **true**, the command uses the design view instead of the frame view for all cells by automatically setting the **signoff.check_drc.read_design_views** application option to **{*}**. If the cell libraries do not contain design views, the command fails.

The **-check_all_runset_layers** and **-select_layers** options are mutually exclusive; you can specify only one.

-select_layers *collection_of_layers*

Specifies the routing layers (metal or via layers) on which to perform design rule checking. The layer names must be the names defined in the technology file.

By default, the command checks the design rules on all routing layers (metal and via layers).

The **-check_all_runset_layers** and **-select_layers** options are mutually exclusive; you can specify only one.

-select_rules ***list_of_rule_names***

Specifies the rules to check. You can specify either the rule name or number. You can specify wildcards. The rule names are declared in the COMMENT option of the foundry runset.

By default, all design rules are checked.

If you specify both the **-select_rules** and **-unselect_rules** options, the **-unselect_rules** option unselects rules from the selected ones.

-unselect_rules ***list_of_rule_names***

Specifies the rules that are not checked. You can specify either the rule name or number. You can specify wildcards. The rule names are declared in the COMMENT option of the foundry runset.

By default, all design rules are checked (no rules are excluded).

If you specify both the **-select_rules** and **-unselect_rules** options, the **-unselect_rules** option unselects rules from the selected ones.

-coordinates ***region_spec***

Specifies the rectangular or rectilinear regions in which to perform design rule checking. You can specify one or more regions.

To specify each rectangular region, use the following syntax to specify the coordinates of the lower-left and upper-right corners of the region:

{ {*llx lly*} {*urx ury*} }

To specify each rectilinear region, use the following syntax to specify the coordinates of the region:

{ {*x1 y1*} {*x2 y2*} ... {*xn yn*} }

The coordinates are floating point numbers in main library units.

If you do not specify this option or the **-excluded_coordinates** option, the command performs design rule checking on the entire block.

If you specify both the **-coordinates** and **-excluded_coordinates** options, the **-excluded_coordinates** option excludes regions from design rule checking based on the selected regions.

This option is mutually exclusive with the **-auto_eco true** option.

-excluded_coordinates ***region_spec***

Specifies the rectangular or rectilinear regions to exclude from design rule checking. You can specify one or more regions.

To specify each rectangular region, use the following syntax to specify the coordinates of the lower-left and upper-right corners of the region:

{ {*llx lly*} {*urx ury*} }

To specify each rectilinear region, use the following syntax to specify the coordinates of the region:

{ {*x1 y1*} {*x2 y2*} ... {*xn yn*} }

The coordinates are floating point numbers in main library units.

If you do not specify this option or the **-coordinates** option, the command performs design rule checking on the entire block.

If you specify both the **-coordinates** and **-excluded_coordinates** options, the **-excluded_coordinates** option excludes regions from design rule checking based on the selected regions.

This option is mutually exclusive with the **-auto_eco true** option.

-auto_eco true | **false**

Controls whether design rule checking is performed only for the automatically detected ECO change area.

By default (**false**), the command performs design rule checking on the entire block.

If **true**, the command compares the block against the version used for the previous run of the **signoff_check_drc** command and detects any physical object changes between them. Design rule checking is performed only for the changed area.

You can use this option only if you have previously run the **signoff_check_drc** command for the current block (or the block specified by the **-pre_eco_design** option).

When you use this option, you cannot use the **-error_data** option. The results of the incremental DRC run are always stored in an error data file named signoff_check_drc_incremental.err. If the **signoff.check_drc.merge_incremental_error_data** application option is set to **true**, the command creates an additional error data file named signoff_check_drc_incremental_merged.err.

The command returns a value of -1 if the incremental DRC run is successful, or if there is no change in the block. If the changes to a block are more than a fixed threshold, the command returns the actual percent change in the block. The default threshold is 20 percent. To specify the threshold value, set the **signoff.check_drc.auto_eco_threshold_value** application option.

This option is mutually exclusive with the **-coordinates** and **-excluded_coordinates** options.

-pre_eco_design ***block_name***

Specifies the block to compare to the current block to determine the ECO change area. You must have previously run a full **signoff_check_drc** command on the specified block.

If you do not specify this option, the tool compares the current block to the version that was used for the previous run of the **signoff_check_drc** command.

This option is valid only with the **-auto_eco true** option.

DESCRIPTION

This command launches the IC Validator tool with a foundry runset to check for DRC violations in the current block. When you use the **signoff_check_drc** command for DRC checking, you must provide cell libraries that contain the child cell data and do not have any DRC violations.

Before running this command, you must set up the environment for the IC Validator tool. For more details about the IC Validator tool, see the *IC Validator Reference Manual*.

The error data generated by the command is not persistent and will be lost if you close the block without saving it.

To reduce the runtime for design rule checking with the IC Validator tool, you can enable multicore processing by using the **set_host_options** command. The IC Validator tool determines the multithreading setting automatically. The **set_host_options** settings are not persistent. You must set them in each session. In addition, the settings are cumulative so that you can define different numbers of jobs on different machines.

After you complete design rule checking on the routing layers, you can perform a quick design rule check on the current block by rerunning the **signoff_check_drc** command with the **-check_all_runset_layers** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the drc.rs runset to perform IC Validator design rule checking. The IC Validator tool reads the design view for the top-level block and all library cell instances and pins only from the frame view for all macro cell instances.

```
prompt> set_app_options -name signoff.check_drc.runset \
  -value "drc.rs"
prompt> signoff_check_drc
```

The following example uses the drc.rs runset to perform IC Validator design rule checking. The IC Validator tool reads the design view for the top-level block, all library cell instances, and all instances of the M1 macro; it reads the frame views with blockages and pins for all other macro cell instances.

```
prompt> set_app_options -name signoff.check_drc.runset \
  -value "drc.rs"
prompt> set_app_options \
  -name signoff.check_drc.read_design_views -value {M1}
prompt> set_app_options \
  -name signoff.ignore_blockages_from_cell -value false
prompt> signoff_check_drc
```

The following example uses the drc.rs runset to perform IC Validator design rule checking. The IC Validator tool reads the design view for the top-level block and all cell instances. The command reports a maximum of 6000 errors per rule.

```
prompt> set_app_options \
  -name signoff.check_drc.read_design_views -value {*}
prompt> set_app_options \
  -name signoff.check_drc.max_errors_per_rule -value 6000
prompt> set_app_options -name signoff.check_drc.runset \
  -value "drc.rs"
prompt> signoff_check_drc
```

The following example skips the DRC violations inside all library cell instances and runs the IC Validator tool in the working directory named **my_signoff_run**.

```
prompt> set_app_options \
  -name signoff.check_drc.ignore_child_cell_errors -value true
prompt> set_app_options -name signoff.check_drc.run_dir \
  -value "my_signoff_run"
prompt> set_app_options -name signoff.check_drc.runset \
  -value "drc.rs"
prompt> signoff_check_drc
```

The following example always reads the fill data. The default behavior is to read the fill data only if its timestamp is newer than the timestamp of the design view.

```
prompt> set_app_options -name signoff.check_drc.fill_view_data \
  -value read
```

```
prompt> set_app_options -name signoff.check_drc.runset \
  -value "drc.rs"
prompt> signoff_check_drc
```

The following example excludes standard cells, macro cells, I/O pad cells, and filler cells from design rule checking, and runs the IC Validator tool with a user-defined option named **-turbo**.

```
prompt> set_app_options \
  -name signoff.check_drc.excluded_cell_types \
  -value {lib_cell macro pad filler}
prompt> set_app_options \
  -name signoff.check_drc.user_defined_options -value "-turbo"
prompt> set_app_options -name signoff.check_drc.runset \
  -value "drc.rs"
prompt> signoff_check_drc
```

The following example runs design rule checking with a customized error data file, **my_error.err**.

```
prompt> signoff_check_drc -error_data {my_error}
```

The following example runs design rule checking on the M1 and M3 layers of the current block.

```
prompt> signoff_check_drc -select_layers { M1 M3 }
```

The following example runs design rule checking for the rules whose names and descriptions match the M*.W.1 or M*.W.3 patterns:

```
prompt> signoff_check_drc -select_rules { M*.W.1*  M*.W.3* }
```

The following example runs design rule checking for all rules in the specified runset, except those whose names and descriptions match the VIA*.S pattern.

```
prompt> signoff_check_drc -unselect_rules { VIA*.S* }
```

The following example runs design rule checking for the rules whose names and descriptions match the M2.A.2 or M4.S.2 patterns:

```
prompt> signoff_check_drc -select_rules { M2.A.2*  M4.S.2* }
```

The following example runs design rule checking for the rules whose names and descriptions match the *.EN.3 pattern:

```
prompt> signoff_check_drc -select_rules { *.EN.3* }
```

The following example runs design rule checking for all rules in the specified runset, except those whose names and descriptions match the *2.EN.*.1 pattern.

```
prompt> signoff_check_drc -unselect_rules { *2.EN.*.1* }
```

The following example runs design rule checking within the rectangle whose lower-left corner is at (22.47, 98.85) and upper-right corner is at (175.39, 187.35).

```
prompt> signoff_check_drc \
  -coordinates { { {22.47 98.85} {175.39 187.35} } }
```

The following example runs design rule checking within the rectilinear region with the following coordinates: (0, 0), (100, 0), (100, 100), (50, 100), (50, 50), and (0, 50).

```
prompt> signoff_check_drc \
  -coordinates {{0 0} {100 0} {100 100} {50 100} {50 50} {0 50}}
```

The following example excludes the rectangular region with its lower-left corner at (50.12, 78.85) and its upper-right corner at (135.43, 157.36) from design rule checking.

```
prompt> signoff_check_drc \
```

-excluded_coordinates { { {50.12 78.85} {135.43 157.36} } }

The following example runs IC Validator using distributed processing on multiple cores of the local machine by starting the IC Validator run with the **-dp4** option.

```
prompt> set_host_options -num_processes 4 {localhost}
prompt> signoff_check_drc
```

The following example runs IC Validator using distributed processing on machine_A and machine_B by starting the IC Validator run with the **-dp -dphosts machine_A machine_A machine_B** options.

```
prompt> set_host_options -num_processes 2 {machine_A}
prompt> set_host_options -num_processes 1 {machine_B}
prompt> signoff_check_drc
```

The following example runs design rule checking for the automatically detected ECO change area.

```
prompt> signoff_check_drc -auto_eco true
```

The following example runs design rule checking for the ECO change area determined by comparing the current block to the block named TOPA.

```
prompt> signoff_check_drc -auto_eco true -pre_eco_design TOPA
```

The following example runs design rule checking for the ECO change area and merges the result from a previous **signoff_check_drc -error_data "pre_eco_drc"** run. The resulting error data file is called signoff_check_drc_incremental_merged.err.

```
prompt> set_app_options \
  -name signoff.check_drc.merge_incremental_data -value true
prompt> set_app_options \
  -name signoff.check_drc.merge_base_error_name \
  -value "pre_eco_drc.err"
prompt> signoff_check_drc -auto_eco true
```

The following example specifies stream files with layout data that replace the layout in the cell libraries for the **signoff_check_drc** run.

```
prompt> set_app_options -name signoff.physical.merge_stream_files \
  -value {stream_file1.gds stream_files2.oas}
prompt> signoff_check_drc
```

SEE ALSO

check_routes(2)
set_host_options(2)
report_host_options(2)
remove_host_options(2)
signoff.check_drc_options(3)
signoff.physical.merge_stream_files(3)

signoff_check_drc_icv_live

Run ICV-Live to performs design rule checking on the current block using a foundry runset.

SYNTAX

signoff_check_drc_icv_live
-rect *{{lx ly} {ux uy}}*
[-layers *layers*]
[-child_depth *number*]

Data Types

lx float
ly float
ux float
uy float
layers list of layer names
number integer

ARGUMENTS

-rect *{{lx ly} {ux uy}}*

Specifies the rectangle area to check.

-layers *layers*

Specifies the layers on which to perform design rule checking. The layer names must be the names defined in the technology file. Default is all routing layers (metal and via layers).

-child_depth *number*

Specifies the maximum hierarchical block depth to process. Default is 0.

DESCRIPTION

This command launches the ICV-Live tool with a foundry runset to check for DRC violations in the current block specified rectangle area.

EXAMPLES

The following example uses the drc.rs runset and layer_map.map file to perform ICV-Live design rule checking.

```
prompt> set_app_options -name signoff.check_drc_live.runset -value "drc.rs"
prompt> set_app_options -name signoff.physical.layer_map_file -value "layer_map.map"
prompt> signoff_check_drc_icv_live -rect {{26.257 24.928} {31.932 21.200}} -layers { M1 VIA1 M2 VIA2 }
```

SEE ALSO

set_signoff_check_drc_icv_live(2)
signoff.check_drc_live.runset(3)
signoff.check_drc_live.run_dir(3)
signoff.check_drc_live.default_layers(3)
signoff.physical.layer_map_file(3)

signoff_create_metal_fill

Invokes the IC Validator tool to insert metal fill to meet the metal density requirements.

SYNTAX

status **signoff_create_metal_fill**
[-mode overwrite | add | remove | replace]
[-select_layers *layers*]
[-all_runset_layers true | false]
[-coordinates *list_of_points*]
[-excluded_coordinates *list_of_points*]
[-nets *nets*]
[-timing_preserve_setup_slack_threshold *threshold*]
[-auto_eco true | false]
[-pre_eco_design *design_name*]
[-track_fill off | generic | *foundry_node*]
[-fill_all_tracks true | false]
[-report_density on | off | *prefix*]
[-output_colored_fill true | false]
[-track_fill_parameter_file auto | generate_file_only | *track_fill_parameter_file_path*]
[-remove_by_rule *rules*]
[-foundry_fill_type both | feol | beol]
[-foundry_for_feol_fill off | list | *foundry_node*]

Data Types

layers	collection
list_of_points	list
nets	collection
threshold	float
design_name	string
foundry_node	string
prefix	string
track_fill_parameter_file_path	string
rules	list

ARGUMENTS

-mode overwrite | add | remove | replace

Specifies the mode of the metal fill operation.

The valid values are

- **overwrite** (the default)
  Replaces the existing metal fill in the current block with new metal fill.

  This mode is supported in both the pattern-based (foundry-qualified runset) and track-based fill flows.

- **add**
  Appends metal fill to the existing metal fill in the current block. The existing fill data is not modified.

  This mode is supported for both pattern-based and track-based fill flows. You can add pattern-based fill on top of pattern-based fill, track-based fill on top of track-based fill, or track-based fill on top of pattern-based fill.

  The main purpose of this mode is to add fill in empty regions. If you run the timing-driven fill flow without fix-density mode enabled, the command removes fill shapes near timing-critical nets, which could create low density pockets. You can use add mode to add fill in these low-density pockets. By default, the command inserts fill to the maximum extent that is allowed. For example, in the track-based fill flow, if you use the sparse fill mode by using the **-fill_all_tracks false** option, the command ensures that the new fill maintains one-track spacing from signal shapes. To limit fill insertion to the low-density pockets, set the **signoff.create_metal_fill.fix_density_errors** application option to **true**. Limiting fill insertion to low-density pockets is supported only when using the track-based fill flow. The density rules must be specified in the technology file or user parameter file. The command uses the density rules to find the density-violating windows.

  You can also use add mode to insert fill in selected regions by using the **-coordinates** option. Use the area-based add mode to insert fill in limited areas, where the combined area is less than 20 percent of the overall chip area. This flow runs on the sparse view of the block and should run at least three times faster than a full-chip run. Because of the limited visibility of data in the sparse view, dense track fill cannot be generated for layers that have double-patterning rules; therefore, the command ignores the **-fill_all_tracks true** option for such layers.

  The command ignores the following application options for track-based fill insertion in add mode:

  - **signoff.create_metal_fill.flat**
  - **signoff.create_metal_fill.max_density_threshold**

- **remove**
  Removes metal fill from the current block.

  By default, the IC Validator tool removes all metal fill from the current block, including the typical critical dimension (TCD) structures. To restrict the metal fill removal, use one or more of the following options:

  - **-select_layers**, which enables layer-based fill removal
  - **-coordinates**, which enables area-based fill removal
  - **-nets**, which removes the metal fill around the shapes of the specified nets based on the spacing requirements specified by the **signoff.create_metal_fill.space_to_nets**, **signoff.create_metal_fill.space_to_clock_nets**, and **signoff.create_metal_fill.space_to_nets_on_adjacent_layer** application options
  - **-timing_preserve_setup_slack_threshold**, which removes the metal fill around the shapes of the nets with a slack less than the specified value based on the spacing requirements specified by the **signoff.create_metal_fill.space_to_nets** and **signoff.create_metal_fill.space_to_nets_on_adjacent_layer** application options
    Note that the tool uses the existing fill when calculating the slack values for pattern-based fill, but not for track-based fill.
  - **-remove_by_rule**, which removes the metal fill based on the specified rules

  This mode is supported in both the pattern-based and track-based flows.

- **replace**
  Replaces the existing metal fill in specific regions of the current block with new metal fill. When you use this mode, you must specify the ECO regions by using at least one of the following options: **-select_layers**, **-coordinates**, **-**

excluded_coordinates, **-nets**, and **-timing_preserve_setup_slack_threshold**.

This mode is supported in both the pattern-based and track-based flows.

-select_layers *layers*

Specifies the routing layers (metal or via layers) on which to perform the metal fill operation. The layer names must be the names defined in the technology file.

If you do not specify this option, the command performs the metal fill operation on all routing layers.

The **-all_runset_layers** and **-select_layers** options are mutually exclusive; you can specify only one.

-all_runset_layers true | false

If **true**, the command inserts fill for all output fill layers defined in the runset. By default (**false**), the command inserts fill only for the routing layers (metal and via layers).

This option is mutually exclusive with the following options: **-select_layers**, **-coordinates**, and **-excluded_coordinates**.

-coordinates *list_of_points*

Specifies the rectangular or rectilinear regions in which to perform the metal fill operation. You can specify one or more regions.

To specify each rectangular region, use the following syntax to specify the coordinates of the lower-left and upper-right corners of the region:

{ {*llx lly*} {*urx ury*} }

To specify each rectilinear region, use the following syntax to specify the coordinates of the region:

{ {*x1 y1*} {*x2 y2*} ... {*xn yn*} }

The coordinates are floating point numbers in main library units.

If you do not specify this option or the **-excluded_coordinates** option, the command performs the metal fill operation on the entire block.

If you specify both the **-coordinates** and **-excluded_coordinates** options, the **-excluded_coordinates** option excludes regions from the metal fill operation based on the selected regions.

This option is mutually exclusive with the **-auto_eco true** option.

-excluded_coordinates *list_of_points*

Specifies the rectangular or rectilinear regions to exclude from the metal fill operation.

To specify each rectangular region, use the following syntax to specify the coordinates of the lower-left and upper-right corners of the region:

{ {*llx lly*} {*urx ury*} }

To specify each rectilinear region, use the following syntax to specify the coordinates of the region:

{ {*x1 y1*} {*x2 y2*} ... {*xn yn*} }

The coordinates are floating point numbers in main library units.

If you do not specify this option or the **-coordinates** option, the command performs the metal fill operation on the entire block.

If you specify both the **-coordinates** and **-excluded_coordinates** options, the **-excluded_coordinates** option excludes regions from the metal fill operation based on the selected regions.

This option is mutually exclusive with the **-auto_eco true** option.

-nets *nets*

Specifies the critical nets around which to preserve timing by preventing fill shapes within the minimum spacing of the net shapes associated with the specified nets.

By default, the minimum spacing between the critical net shapes and the fill shapes is based on the minimum spacing specified in the technology file.

- For fill on the same layer as the net shape, the default minimum spacing is two times the minimum spacing value specified for the layer in the technology file.

  To explicitly specify the same layer minimum spacing requirements, set the **signoff.create_metal_fill.space_to_nets** application option.

  To explicitly specify the same layer minimum spacing requirements for clock nets, set the **signoff.create_metal_fill.space_to_clock_nets** application option.

  To add the fill shapes till the clock net shields, set the **signoff.create_metal_fill.fill_shielded_clock** application option.

- For fill in the vertical extension of the net shape on the adjacent layers, the default minimum spacing is the minimum spacing value specified for the layer in the technology file.

  To explicitly specify the adjacent layer minimum spacing requirements, set the **signoff.create_metal_fill.space_to_nets_on_adjacent_layer** application option.

  To explicitly specify the adjacent layer minimum spacing requirements for clock nets, set the **signoff.create_metal_fill.space_to_clock_nets_on_adjacent_layer** application option.

  To allow metal fill in the vertical extension of the net shape on the adjacent layers, set the **signoff.create_metal_fill.fill_over_net_on_adjacent_layer** application option to **true**. In this case, the command does not use the adjacent-layer minimum spacing value.

-timing_preserve_setup_slack_threshold *threshold*

Specifies the setup slack threshold in library time units. The command considers all nets with slack less than the specified threshold as critical nets and preserves the timing for these nets by preventing fill shapes within the minimum spacing of the net shapes associated with these critical nets.

By default, the minimum spacing between the critical net shapes and the fill shapes is based on the minimum spacing specified in the technology file.

- For fill on the same layer as the net shape, the default minimum spacing is two times the minimum spacing value specified for the layer in the technology file.

  To explicitly specify the same layer minimum spacing requirements, set the **signoff.create_metal_fill.space_to_nets** application option.

- For fill in the vertical extension of the net shape on the adjacent layers, the default minimum spacing is the minimum spacing value specified for the layer in the technology file.

  To explicitly specify the adjacent layer minimum spacing requirements, set the **signoff.create_metal_fill.space_to_nets_on_adjacent_layer** application option.

  To allow metal fill in the vertical extension of the net shape on the adjacent layers, set the **signoff.create_metal_fill.fill_over_net_on_adjacent_layer** application option to **true**. In this case, the command does not use the adjacent-layer minimum spacing value.

The default threshold is null, which means that the tool does not automatically determine the critical nets.

You cannot use this option with the **-select_layers** option.

-auto_eco true | **false**

Performs fill removal and refill in the automatically detected ECO change areas. You can use this option only if you have previously run the **signoff_create_metal_fill** command.

By default, when you use the **-auto_eco true** option, the tool compares the current block against the version used for the previous run of the **signoff_create_metal_fill** command and detects any physical object changes between them. To compare the current block to a different block, use the **-pre_eco_design** option to specify the comparison block.

You can run this command immediately after an ECO change to remove overlapped metal fill and keep the density consistent by refilling empty areas created during the ECO change. The command performs fill removal and refill only on the layers with automatically detected changes. If a cell instance changes, the command takes a conservative approach and considers changes on all layers. You can use this option multiple times after every ECO change to do a fast refill in the ECO areas.

Note that a change on a metal layer also triggers fill removal and refill on the adjacent via layers to ensure that no floating or hanging vias remain after the metal shapes are removed from the ECO area.

You can use this option with the **-nets** and **-timing_preserve_setup_slack_threshold** options to ensure that timing is preserved in the ECO area during refill.

If you use the **-select_layers** option, the specified layers must be a subset of the automatically detected changed layers. Any extra layers are ignored. For example, if the automatically detected changed layers are M2, M3, and V2, you can choose to make changes only the M2 layer by using the **-select_layers {M2}** option.

This option is mutually exclusive with the **-mode**, **-all_runset_layers**, **-coordinates**, and **-excluded_coordinates** options.

If the run is successful or there are no changes in the block, the command returns a value of 1. If the percentage of change in the block is greater than the change threshold, the command does not perform metal fill insertion and returns the percentage of change in the design. The default change threshold is 20 percent; to modify the change threshold, set the **signoff.create_metal_fill.auto_eco_threshold_value** application option.

-pre_eco_design ***design_name***

Specifies the block to compare to the current block to determine the ECO change area. You must have previously run a full **signoff_create_metal_fill** command on the specified block.

If you do not specify this option, the tool compares the current block to the version that was used for the previous run of the **signoff_create_metal_fill** command.

This option is valid only with the **-auto_eco** option.

-track_fill off | **generic** | ***foundry_node***

Controls whether the command uses the pattern-based or track-based flow.

By default (**off**), the command uses the pattern-based flow with a user-specified runset.

If you enable the track-based flow by setting this option to **generic** or a *foundry_node* keyword, the tool inserts track-based fill with an automatically generated runset. The track-based flow does not use a user-specified runset.

The *foundry_node* keywords correspond to specific foundries and technology nodes. To see a list of the valid values, use the **-track_fill list** option. For an example, see the EXAMPLES section.

By default, track-based metal fill insertion uses sparse mode, and skips one track between signal shapes and fill shapes. To use dense mode, which does not skip tracks, set the **-fill_all_tracks** option to **true**.

If you are using generic track-based metal fill insertion on a design with double-patterning technology, you must use sparse mode unless the design is precolored.

-fill_all_tracks true | false

Controls whether track-based metal fill insertion uses sparse mode or dense mode.

By default (**false**), track-based metal fill insertion uses sparse mode and skips a track between the signal shapes and the fill shapes.

If **true**, track-based metal fill insertion uses dense mode and does not skip tracks between the signal shapes and the fill shapes. If you enable dense mode and the block uses double-patterning technology, the tool considers the mask constraints during metal fill insertion and colors the fill shapes to avoid double-patterning violations. If the block is not precolored, runtime increases because the IC Validator tool must first color the block.

This option is valid only for the track-based flow.

-report_density on | off | *prefix*

Controls whether the command creates a detailed density and density gradient report.

By default (**off**), the command does not generate density reports.

When set to **on**, the command generates density reports with the following names:
track_fill_color_balance_and_density_report.txt and track_fill_density_gradient_report.txt.

To replace "track_fill" with a user-specified string, specify the string as the *prefix* argument. In this case, the command generates the following reports: *prefix*_color_balance_and_density_report.txt and *prefix*_density_gradient_report.txt.

This option is valid only for the track-based flow.

-output_colored_fill true | false

Controls whether the command sets mask constraints on the fill shapes inserted on double-patterning layers.

By default (**false**), the command does not set mask constraints on the fill shapes.

When **true**, the command assigns a data type of 235 for fill shapes with a **mask_one** mask constraint and 236 for fill shapes with a **mask_two** mask constraint. To assign different data types for the colored fill, set the following parameters in the IC Validator parameter file: **mx_fill_datatype_color1**, **viax_fill_datatype_color1**, **mx_fill_datatype_color2**, and **viax_fill_datatype_color2**.

This option is valid only for the track-based flow.

-track_fill_parameter_file auto | generate_file_only | *track_fill_parameter_file_path*

Specifies behavior related to the IC Validator parameter file. Valid values are

- **auto** (the default)
  The tool generates a parameter file with default values. It uses this file for the metal fill operation and saves it in the fill run directory as track_fill_params.rh.
- **generate_file_only**
  The tool generates a parameter file with default values and saves it in the fill run directory as track_fill_params.rh. When you specify this value, the command quits after generating the parameter file and does not perform the metal fill operation.
- *track_fill_parameter_file_path*
  Specifies the parameter file to use for the metal fill operation.

  To generate the parameter file, you can start with the automatically generated parameter file and customize the settings as needed.

This option is valid only for the track-based flow.

-remove_by_rule {*rules*}

Specifies the type of rule-based fill removal.

By default (**off**), the command does not perform rule-based fill removal.

To enable rule-based fill removal, specify one or more of the following values:

- **max_density_threshold**
  Fill removal honors the maximum density rules specified by the **signoff.create_metal_fill.max_density_threshold** application option.

  This rule is supported only in the track-based flow.

- **ndr**
  Fill removal honors the nondefault routing rules set on the nets.

  This rule is supported in both the pattern-based and track-based flows.

-foundry_fill_type both | feol | beol

Specifies the type of fill to be done using **signoff_create_metal_fill** command on metal layers or base metal layers.

By default, the command does foundry fill on metal layers.

When the option is set to \fboth, the command does the subsequent foundry fill on metal layers and then on base metal layers.

When the option is set to \fbeol, the command does the foundry fill only on metal layers.

When the option is set to \ffeol, the command does the foundry fill only on base metal layers.

-foundry_for_feol_fill off | list | *foundry_node*

Controls whether the command uses the marker layers for base metal layers. The marker layers are chose based on foundry node.

By default (**off**), the command uses no marker layers for base metal layers.

The *foundry_node* keywords correspond to specific foundries and technology nodes. To see a list of the valid values, use the **-foundry_for_feol_fill list** option. For an example, see the EXAMPLES section.

DESCRIPTION

This command invokes the IC Validator tool to perform metal fill operations on the current block. You can use this command to

- Insert metal fill to meet the metal density requirements
- Perform post ECO metal fill removal and re-fill
- Remove metal fill

  The command can use either a pattern-based flow with a user-specified runset or a track-based flow with an automatically generated runset.

  After performing the metal fill operation, the command saves the block. The metal fill data is stored as an internal fill design that consists of the area-fill objects.

  Before using this command,

  - The block must be fully routed design with little or no DRC violations.

- You must save the block to disk (the IC Validator tool reads the design from disk).
- You must define the environment settings for the IC Validator tool.
  For details about the IC Validator tool, see the *IC Validator Reference Manual*.

  To reduce the runtime of the IC Validator tool, you can enable multicore processing by using the **set_host_options** command. The IC Validator tool determines the multithreading setting automatically. Note that the **set_host_options** settings are not persistent. They must be set in each session. In addition, the setting is cumulative so that different numbers of jobs can be defined on different machines.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example performs pattern-based fill insertion on the entire block using a runset named fill.rs.

```
prompt> set_app_options \
  -name signoff.create_metal_fill.runset -value "fill.rs"
prompt> signoff_create_metal_fill
```

The following example fills all empty regions with flattened metal fill.

```
prompt> set_app_options \
  -name signoff.create_metal_fill.runset -value "fill.rs"
prompt> set_app_options \
  -name signoff.create_metal_fill.flat -value true
prompt> signoff_create_metal_fill
```

The following example fills all empty regions on the metal1 layer outside the region {{100 150} {300 200}}.

```
prompt> set_app_options \
  -name signoff.create_metal_fill.runset -value "fill.rs"
prompt> signoff_create_metal_fill -select_layers {metal1} \
  -excluded_coordinates {{{100 150} {300 200}}}
```

The following example removes the internal fill design from the current block.

```
prompt> signoff_create_metal_fill -mode remove
```

The following example removes the fill from the metal1 and metal3 layers in the internal fill design of the current block.

```
prompt> signoff_create_metal_fill -mode remove \
  -select_layers {metal1 metal3}
```

The following example removes track fill around the n1 and n2 nets.

```
prompt> signoff_create_metal_fill -track_fill foundryNode \
  -mode remove -nets {n1 n2}
```

The following example removes all metal fill on the metal1 and metal3 layers, and then refills those two metal layers. The existing metal fill on other metal layers remains unchanged.

```
prompt> set_app_options \
  -name signoff.create_metal_fill.runset -value "fill.rs"
```

```
prompt> signoff_create_metal_fill -mode replace \
  -select_layers {metal1 metal3}
```

The following example fills all empty regions in {{100 150} {300 200}} without overwriting the existing metal fill.

```
prompt> set_app_options \
  -name signoff.create_metal_fill.runset -value "fill.rs"
prompt> signoff_create_metal_fill -mode add \
  -coordinates {{{100 150} {300 200}}}
```

The following example re-creates track fill for layer M4. Timing and density-based rules are honored during fill replacement. Because M4 is replaced, VIA3 and VIA4 are automatically replaced.

```
prompt> set_app_options \
  -name signoff.create_metal_fill.space_to_nets \
  -value {{M2 4x} {M3 4x} {M4 4x} {M5 5x}}
prompt> set_app_options \
  -name signoff.create_metal_fill.fix_density_errors -value true
prompt> set_app_options \
  -name signoff.create_metal_fill.max_density_threshold \
  -value {{M4 30}}
prompt> signoff_create_metal_fill -track_fill foundryNode \
  -mode replace -select_layers {M4} \
  -timing_preserve_setup_slack_threshold 0.5
```

The following example uses the track-based flow to re-create the via fill for the VIA1, VIA2, VIA3, VIA4, and VIA5 via layers.

```
prompt> signoff_create_metal_fill -track_fill foundryNode \
  -mode replace -select_layers {VIA1 VIA2 VIA3 VIA4 VIA5}
```

The following example fills all empty regions in the rectilinear polygon {{0 0} {100 0} {100 100} {50 100} {50 50} {0 50}} without overwriting the existing metal fill.

```
prompt> set_app_options \
  -name signoff.create_metal_fill.runset -value "fill.rs"
prompt> signoff_create_metal_fill -mode add -track_fill foundryNode \
  -coordinates {{0 0} {100 0} {100 100} {50 100} {50 50} {0 50}}
```

The following example adds fill in empty regions without modifying the existing fill and maintains sufficient spacing from the critical nets n1, n2, and n3.

```
prompt> signoff_create_metal_fill -track_fill foundryNode \
  -mode add -nets "n1 n2 n3"
```

The following example adds track fill only in density-violation windows without modifying existing fill.

```
prompt> set_app_options \
  -name signoff.create_metal_fill.fix_density_errors -value true
prompt> signoff_create_metal_fill -track_fill foundryNode \
  -mode add -fill_all_tracks true
```

The following example adds track fill only in density-violation windows without modifying existing fill. Within the density-violating windows, the command tries to add fill shapes away from the critical nets n1, n2, and n3.

```
prompt> set_app_options \
  -name signoff.create_metal_fill.fix_density_errors -value true
prompt> signoff_create_metal_fill -track_fill foundryNode \
  -mode add -fill_all_tracks true -nets "n1 n2 n3"
```

The following example invokes IC Validator with user-defined options named **-turbo** and **-vue**.

```
prompt> set_app_options \
  -name signoff.create_metal_fill.runset -value "fill.rs"
prompt> set_app_options \
  -name signoff.create_metal_fill.user_defined_options \
  -value "-turbo -vue"
prompt> signoff_create_metal_fill
```

The following example lists all supported foundry nodes for the track-based fill flow.

```
prompt> signoff_create_metal_fill -track_fill list
```

The following example creates track-based fill for foundryNode by creating sparse fill to overwrite the existing metal fill.

```
prompt> signoff_create_metal_fill -track_fill foundryNode
```

The following example creates track-based fill for the generic process node by creating sparse fill to overwrite the existing metal fill.

```
prompt> signoff_create_metal_fill -track_fill generic
```

The following example creates track-based fill for foundryNode by creating dense fill to overwrite the existing metal fill.

```
prompt> signoff_create_metal_fill -track_fill foundryNode \
  -fill_all_tracks true
```

The following example creates dense track-based fill for foundryNode. It attaches mask attributes to the fill shapes and outputs the colored fill using datatype 235 for color1 and 236 for color2.

```
prompt> signoff_create_metal_fill -track_fill foundryNode \
  -fill_all_tracks true -output_colored_fill true
```

The following example creates sparse track-based fill for foundryNode and generates density reports with the following names: sparse_fill_color_balance_and_density_report.txt and sparse_fill_density_gradient_report.txt.

```
prompt> signoff_create_metal_fill -track_fill foundryNode \
  -report_density "sparse_fill"
```

The following example creates sparse track-based fill for foundryNode node using a customized track fill parameter file named custom_params.rh.

```
prompt> signoff_create_metal_fill -track_fill foundryNode \
  -track_fill_parameter_file ./custom_params.rh
```

The following example honors the nondefault routing rules during fill removal.

```
prompt> signoff_create_metal_fill -track_fill foundryNode \
  -mode remove -remove_by_rule {ndr}
```

The following example honors the maximum density rules during fill removal, so it reduces the metal density of the M2 and M3 layers to close to 30 percent.

```
prompt> set_app_options \
  -name signoff.create_metal_fill.max_density_threshold \
  -value {{M2 30} {M3 30}}
prompt> signoff_create_metal_fill -track_fill foundryNode \
  -mode remove -remove_by_rule {max_density_threshold}
```

The following example runs the IC Validator tool using distributed processing on multiple cores of the local machine by starting the IC Validator run with the **-dp4** option.

```
prompt> set_host_options -num_processes 4 {localhost}
prompt> set_app_options \
```

```
    -name signoff.create_metal_fill.runset -value "fill.rs"
  prompt> signoff_create_metal_fill
```

The following example runs the IC Validator tool using distributed processing on machine_A and machine_B by starting the IC Validator run with the **-dp -dphosts machine_A machine_A machine_B** options.

```
  prompt> set_host_options -num_processes 2 {machine_A}
  prompt> set_host_options -num_processes 1 {machine_B}
  prompt> set_app_options \
    -name signoff.create_metal_fill.runset -value "fill.rs"
  prompt> signoff_create_metal_fill
```

The following example performs fill removal and refill for the automatically detected ECO change area. It also makes sure that the newly created fill in the ECO region maintains four times the minimum spacing away from the net1 and net2 nets for the M2 and M3 metal layers.

```
  prompt> set_app_options \
    -name signoff.create_metal_fill.runset -value "fill.rs"
  prompt> set_app_options \
    -name signoff.create_metal_fill.space_to_nets \
    -value {{M2 4x} {M3 4x}}
  prompt> signoff_create_metal_fill -nets "net1 net2" -auto_eco true
```

The following example runs incremental fill for the ECO change area determined by comparing the current block to the block named TOPA.

```
  prompt> set_app_options \
    -name signoff.create_metal_fill.runset -value "fill.rs"
  prompt> signoff_create_metal_fill -auto_eco true \
    -pre_eco_design TOPA
```

The following example specifies stream files with layout data that replace the layout in the cell libraries for the **signoff_create_metal_fill** run.

```
  prompt> set_app_options \
    -name signoff.create_metal_fill.runset -value "fill.rs"
  prompt> set_app_options \
    -name signoff.physical.merge_stream_files \
    -value {stream_file1.gds stream_files2.oas}
  prompt> signoff_create_metal_fill
```

The following example fills all empty regions on poly and diffusion layers.

```
  prompt> set_app_options \
    -name signoff.create_metal_fill.base_metal_layers \
    -value {base_metal_fill.rs}
  prompt> signoff_create_metal_fill -foundry_fill_type feol
```

The following example fills all empty regions on poly and diffusion layers with marker layer fills on particular foundry.

```
  prompt> set_app_options \
    -name signoff.create_metal_fill.base_metal_layers \
    -value {base_metal_fill.rs}
  prompt> signoff_create_metal_fill -foundry_fill_type feol \
  -foundry_for_feol_fill <foundry_node>
```

SEE ALSO

set_host_options(2)
report_host_options(2)
remove_host_options(2)
signoff.create_metal_fill_options(3)
signoff.physical.merge_stream_files(3)

signoff_create_pg_augmentation

Invokes the IC Validator tool to augment existing PG structure for improving IR drop.

SYNTAX

status **signoff_create_pg_augmentation**
[-node tsmc7 | tsmc16 | samsung7 | samsung10]
[-mode add | remove]
[-nets *nets*]
[-timing_preserve_setup_slack_threshold *threshold*]
[-coordinates_ground *list_of_points*]
[-excluded_coordinates_ground *list_of_points*]
[-coordinates_power *list_of_points*]
[-excluded_coordinates_power *list_of_points*]
[-parameter_file auto | *pga_parameter_file_path*]

Data Types

list_of_points	list
nets	collection
pga_parameter_file_path	string

ARGUMENTS

-node tsmc7 | tsmc16 | samsung7 | samsung10

Specifies the node for which PG-Augmentation flow should be run.

The supported nodes are:

- **tsmc7**
- **tsmc16**
- **samsung7**
- **samsung10**

-mode add | remove

Specifies the mode of the pg augmentation operation.

The valid values are

- **add**
  Appends pg augmentation to the existing pg augmentation in the current block. The existing data is not modified.

The main purpose of this mode is to add PGA in empty regions OR for different power nets.

- **remove**
  Removes pg augmentation from the current block.

  By default, the tool removes all pg augmentation from the current block.

-coordinates_ground *list_of_points*

Specifies the rectangular or rectilinear regions in which to perform the pg augmentation operation for ground net. You can specify one or more regions.

To specify each rectangular region, use the following syntax to specify the coordinates of the lower-left and upper-right corners of the region:

{ {*llx lly*} {*urx ury*} }

To specify each rectilinear region, use the following syntax to specify the coordinates of the region:

{ {*x1 y1*} {*x2 y2*} ... {*xn yn*} }

The coordinates are floating point numbers in main library units.

If you specify both the **-coordinates_ground** and **-excluded_coordinates_ground** options, the **-excluded_coordinates_ground** option excludes regions from the pg augmentation operation based on the selected regions.

-excluded_coordinates_ground *list_of_points*

Specifies the rectangular or rectilinear regions to exclude from the pg augmentation operation.

To specify each rectangular region, use the following syntax to specify the coordinates of the lower-left and upper-right corners of the region:

{ {*llx lly*} {*urx ury*} }

To specify each rectilinear region, use the following syntax to specify the coordinates of the region:

{ {*x1 y1*} {*x2 y2*} ... {*xn yn*} }

The coordinates are floating point numbers in main library units.

If you specify both the **-coordinates_ground** and **-excluded_coordinates_ground** options, the **-excluded_coordinates_ground** option excludes regions from the pg augmentation operation based on the selected regions.

-coordinates_power *list_of_points*

Specifies the rectangular or rectilinear regions in which to perform the pg augmentation operation for power net. You can specify one or more regions.

To specify each rectangular region, use the following syntax to specify the coordinates of the lower-left and upper-right corners of the region:

{ {*llx lly*} {*urx ury*} }

To specify each rectilinear region, use the following syntax to specify the coordinates of the region:

{ {*x1 y1*} {*x2 y2*} ... {*xn yn*} }

The coordinates are floating point numbers in main library units.

If you specify both the **-coordinates_power** and **-excluded_coordinates_power** options, the **-excluded_coordinates_power** option excludes regions from the pg augmentation operation based on the selected regions.

-excluded_coordinates_power *list_of_points*

Specifies the rectangular or rectilinear regions to exclude from the pg augmentation operation.

To specify each rectangular region, use the following syntax to specify the coordinates of the lower-left and upper-right corners of the region:

{ {*llx lly*} {*urx ury*} }

To specify each rectilinear region, use the following syntax to specify the coordinates of the region:

{ {*x1 y1*} {*x2 y2*} ... {*xn yn*} }

The coordinates are floating point numbers in main library units.

If you specify both the **-coordinates_power** and **-excluded_coordinates_power** options, the **-excluded_coordinates_power** option excludes regions from the pg augmentation operation based on the selected regions.

-nets *nets*

Specifies the critical nets around which to preserve timing by preventing PGA shapes within the minimum spacing of the net shapes associated with the specified nets.

-timing_preserve_setup_slack_threshold *threshold*

Specifies the setup slack threshold in library time units. The command considers all nets with slack less than the specified threshold as critical nets and preserves the timing for these nets by preventing PGA shapes within the minimum spacing of the net shapes associated with these critical nets.

-parameter_file auto | *pga_parameter_file_path*

Specifies behavior related to the IC Validator parameter file. Valid values are

- **auto** (the default)
  The tool generates a parameter file with default values. It uses this file for the pg augmentation operation.
- *pga_parameter_file_path*
  Specifies the parameter file to use for the pg augmentation operation.

  To generate the parameter file, you can start with the automatically generated parameter file and customize the settings as needed.

DESCRIPTION

This command invokes the IC Validator tool to perform pg augmentation operations on the current block. You can use this command to

- Insert pg augmentation to improve IR drop
- Remove pg augmentation

  After performing the pg augmentation operation, the command saves the block.

  Before using this command,

  - The block must be fully routed design with little or no DRC violations.
  - You must save the block to disk (the IC Validator tool reads the design from disk).

- You must define the environment settings for the IC Validator tool.
  For details about the IC Validator tool, see the *IC Validator Reference Manual*.

  To reduce the runtime of the IC Validator tool, you can enable multicore processing by using the **set_host_options** command. The IC Validator tool determines the multithreading setting automatically. Note that the **set_host_options** settings are not persistent. They must be set in each session. In addition, the setting is cumulative so that different numbers of jobs can be defined on different machines.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example performs pg augmentation insertion on the entire block.

```
prompt> set_app_options \
  -name signoff.create_pg_augmentation.ground_net_name -value "VSS"
prompt> set_app_options \
  -name signoff.create_pg_augmentation.power_net_name -value "VDD"
prompt> signoff_create_pg_augmentation -node tsmc7
```

SEE ALSO

set_host_options(2)
report_host_options(2)
remove_host_options(2)
options(3)

signoff_fix_drc

Invokes router to fix selected signoff DRC violations based on the results from a prior IC Validator run and then reruns IC Validator signoff DRC checking.

SYNTAX

status **signoff_fix_drc**
[-start_repair_loop *start_loop*]
[-max_number_repair_loop *max_loops*]
[-select_rules {*list_of_rule_names*}]
[-unselect_rules {*list_of_rule_names*}]
[-coordinates {*list of bounding box(es)*}]
[-excluded_coordinates {*list of exclude bounding box(es)*}]
[-nets {*timing critical nets*}]
[-timing_preserve_setup_slack_threshold {*float value*}]

Data Types

start_loop integer
end_loop integer
list_of_rule_names list

ARGUMENTS

-start_repair_loop *start_loop*

Specifies the starting repair loop. The repair loops are run from the starting repair loop to the maximum number of repair loop specified by the **-max_number_repair_loop** option.

The range is from 1 to 10. The starting repair loop must be less than or equal to the maximum number repair loop.

The default value is 1.

For example, you can run with the default flow (loops 1 and 2) and then follow up with additional "unique" repair loops, such as 3 and 4, by running the command again with a new starting and maximum number repair loop.

-max_number_repair_loop *max_loops*

Specifies the maximum number of repair loop. DRC fixing ends after the maximum number repair loop or when it detects that there are no more signoff DRC violations left, whichever occurs first.

The range is from 1 to 10. The maximum number repair loop must be greater than or equal to the starting repair loop.

The default value is 2.

For example, you can run with the default flow (loops 1 and 2) and then follow up with additional "unique" repair loops, such as 3

and 4, by running the command again with a new starting and maximum number repair loop.

-select_rules {*list_of_rule_names*}

Specifies the rules to check. You can specify either the rule name or number. You can specify wildcards. The rule names are declared in the COMMENT option of the foundry runset.

By default, all design rules are checked.

If you specify both the **-select_rules** option and the **-unselect_rules** option, the **-unselect_rules** option unselects rules from the selected ones.

-unselect_rules {*list_of_rule_names*}

Specifies the rules that are not checked. You can specify either the rule name or number. You can specify wildcards. The rule names are declared in the COMMENT option of the foundry runset.

By default, all design rules are checked (no rules are excluded).

If you specify both the **-select_rules** option and the **-unselect_rules** option, the **-unselect_rules** option unselects rules from the selected ones.

-coordinates {*list of bounding box(es) from where DRC errors will be fixed*}

Specifies the list of bounding box(es) from where DRC errors will be fixed. The following is the syntax:

{{{llx1 lly1} {urx1 ury1}} ...}

The following example shows the rectilinear area (22.47 98.85) (175.39 187.35) where DRC errors will be fixed prompt> signoff_fix_drc -coordinates { { {22.47 98.85} {175.39 187.35} } }

The following example shows the rectilinear area {{0 0} {100 0} {100 100} {50 100} {50 50} {0 50}} where DRC errors will be fixed. prompt> signoff_fix_drc -coordinates {{0 0} {100 0} {100 100} {50 100} {50 50} {0 50}}

-excluded_coordinates {*list of bounding box(es) from where DRC errors will not be fixed*}

Specifies the list of bounding box(es) from where DRC errors will not be fixed. The following is the syntax:

{{{llx1 lly1} {urx1 ury1}} ...}

The following example shows the rectilinear area (22.47 98.85) (175.39 187.35) where DRC errors will not be fixed. prompt> signoff_fix_drc -excluded_coordinates { { {22.47 98.85} {175.39 187.35} } }

-nets {*list of timing critical nets*}

Specifies the list of timing critical nets that would be protected during fixing flow.

The following example treats nets n1 and n2 as critical during the fixing flow. prompt> signoff_fix_drc -nets {n1, n2}

-timing_preserve_setup_slack_threshold {*float value representing setup slack threshold*}

Specifies the setup slack threshold value for identification of critical nets. The option finds all the paths that have less slack than the specified value and collects the nets that are part of the paths. These timing critical nets are protected during fixing flow.

The following example shows timing will be honored during the fixing flow. Paths that have slack values less than 0.5 are treated as critical prompt> signoff_fix_drc -timing_preserve_setup_slack_threshold 0.5

DESCRIPTION

This command provides tight integration between the Fusion Compiler tool and the IC Validator tool for postroute signoff DRC fixing. It assumes that the current block is near timing closure and there are only a small number of DRC violations left unfixed after the place and route stage.

The command invokes router to fix selected signoff DRC violations based on the results from a prior IC Validator run and then reruns IC Validator signoff DRC checking.

Before you run this command, you must provide the environment settings for IC Validator by using the **set_app_options** command and run the **signoff_check_drc** command or otherwise generate the required IC Validator database. If no database is specified **signoff_fix_drc** will run the signoff_check_drc internally to generate the IC Validator database.

The **set_app_options** for the **signoff_check_drc** command will be applied during the **signoff_fix_drc** command.

Note: When the option "signoff.fix_drc.target_clock_nets" is set to false, the router does NOT have big routing topology changes on clock nets, but it does allow minor shape changes in local area for fixing DRC violations.

To reduce the runtime for signoff autofix DRC flow , you can enable multicore processing by using the **set_host_options** command. router supports multithreading on a multicore machine. IC Validator supports the following types of multicore processing: job submission by a user-specified distributed processing script, job submission by using the Load Sharing Facility (LSF) or Oracle Grid Engine, and distributed processing on the specified hosts. IC Validator determines the multithreading setting automatically. Note that the **set_host_options** settings are not persistent. They must be set in each session. In addition, the setting is cumulative so that different numbers of jobs can be defined on different machines.

Warning

The signoff autofix DRC flow does not address DRC violations within PG nets, clock nets, shielded nets, and frozen nets.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows the signoff autofix DRC flow:

```
prompt> signoff_check_drc
prompt> set_app_options -name signoff.fix_drc.init_drc_error_db \
  -value signoff_check_drc_run
prompt> signoff_fix_drc
```

The following example shows the signoff autofix DRC flow using the design views of all child cells in **signoff_check_drc**:

```
prompt> set_app_options -name signoff.check_drc.read_design_views \
  -value {*}
prompt> signoff_check_drc
prompt> set_app_options -name signoff.fix_drc.init_drc_error_db \
  -value signoff_check_drc_run
prompt> signoff_fix_drc
```

The following example runs router with 4 threads and runs IC Validator using distributed processing on 8 cores of the local machine by starting the IC Validator run with the **-dp32** option.

```
prompt> set_host_options -max_cores 4 -num_process 8
prompt> set_app_options -name signoff.fix_drc.init_drc_error_db \
  -value signoff_check_drc_run
```

```
prompt> signoff_fix_drc
```

The following example runs router with 4 threads and runs IC Validator using distributed processing on 4 cores of the local machine by starting the IC Validator run with the **-dp4** option.

```
prompt> set_host_options -max_cores 4
prompt> set_app_options -name signoff.fix_drc.init_drc_error_db \
  -value signoff_check_drc_run
prompt> signoff_fix_drc
```

Note that the command uses the **-max_cores** setting for both router multithreading and the IC Validator **-dp** setting if and only if the **-max_cores** option is specified alone. The **-max_cores** option has no effect on the IC Validator **-dp** option when it is specified with any other options.

The following example runs the router with one thread and runs IC Validator using distributed processing on eight cores of the local machine by starting the IC Validator run with the **-dp8** option.

```
prompt> set_host_options -num_process 8
prompt> set_app_options -name signoff.fix_drc.init_drc_error_db \
  -value signoff_check_drc_run
prompt> signoff_fix_drc
```

SEE ALSO

set_host_options(2)
report_host_options(2)
remove_host_options(2)
signoff_check_drc(2)

signoff_fix_isolated_via

Identifies and fixes isolated vias in a design by inserting additional vias in the isolated via region.

SYNTAX

status **signoff_fix_isolated_via**
[-check_only true | false]
[-error_data *errdata_name*]
[-save_design true | false]
[-update_track_fill true | false]
[-track_fill_runset_include_file track_fill_runset_include_file_path]

ARGUMENTS

-check_only true | false

Controls whether to check for isolated via violations only or to check and also fix the violations. By default, this command checks for isolated via violations and fixes the violations.

The default is **false**.

-error_data *errdata_name*

Specifies the name of the error data.

If you do not specify this option, the error data is called *signoff_fix_isolated_via*.err.

-save_design true | false

Specifies to save the design with isolated via fixes automatically. By default, the user needs to run "save_block" after command execution to save the design with the isolated via fixes.

The default is **false**.

-update_track_fill true | false

Specifies to run isolated via fixing after track fill. When this option is set to true, signoff_fix_isolated_via will read existing track fill design, identify isolated vias. Isolated via fixes will only be made in the existing track fill design. The main design will not be touched. By default, the signoff_fix_isolated_via flow will ignore the fill design completely and will make fixes in main design.

You must also specify the track_fill_runset_include_file option when the -update_track_fill is specified.

The default is **false**.

-track_fill_runset_include_file track_fill_runset_include_file_path

Specifies the track fill runset include file. The user can find the track_fill_runset_include_file in track fill run directory. This option is

required when the -update_track_fill is specified.

DESCRIPTION

This command invokes the IC Validator tool using an automatically generated runset to identify and fix isolated vias in a design by inserting additional vias in the isolated via region.

Before you use this command, you must open the design library and design view. By default, the opened design view will be chosen to create isolated via fixes.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example runs the command to only check isolated vias in the design view.

```
prompt> signoff_fix_isolated_via \
  -check_only true
```

The following example runs the command to check and fix isolated vias and will save design view with isolated via fixes during the execution of signoff_fix_isolated_via.

```
prompt> signoff_fix_isolated_via \
  -save_design true
```

The following example runs the command to check and fix isolated vias. Furthermore, the command will read track fill design, will identify isolated vias and will create isolated via fixes in track fill design. Please note that with these settings, the main design will not be touched. Also, when you specify -update_track_fill true, you must to specify track_fill_runset_include_file.

```
prompt> signoff_fix_isolated_via \
  -update_track_fill true
  -track_fill_runset_include_file ./user_defined_params.rh
```

signoff_report_metal_density

Reports the metal density and density gradient values.

SYNTAX

status **signoff_report_metal_density**
[-select_layers {*layers*}]
[-coordinates {*list_of_points*}]
[-output {*filename*}]
[-starting_point {*list_of_points*}]

Data Types

layers	collection
list_of_points	list
filename	string

ARGUMENTS

-select_layers {*layers*}

Specifies the routing layers (metal layers) for which to calculate the density and gradient values. The layer names must be the names defined in the technology file.

If you do not specify this option, the command will calculate the density and density gradient values for routing layers in min/max range.

-coordinates {*list_of_points*}

Specifies the rectangular or rectilinear regions in which to calculate the density and gradient values.

To specify each rectangular region, use the following syntax to specify the coordinates of the lower-left and upper-right corners of the region:

{ {*llx lly*} {*urx ury*} }

To specify each rectilinear region, use the following syntax to specify the coordinates of the region:

{ {*x y*} {*x y*} {*x y*} {*x y*} ... }

The units of the coordinates are the same as the main library units.

If you do not specify this option, the command calculate the density and gradient values for the entire block.

-output {*filename*}

To specify the name of density and gradient report output in text format. To specify name of the output file, use the following

syntax:

-output filename.txt

If you do not specify this option, the default filename for density value will be "report_metal_density.txt", and for density gradient "gradient_density_report.txt".

-starting_point {*list_of_points*}

By default, starting point is taken as the lower left corner of bounding box of chip-boundary, from where density and gradient window are created. This option is provided so that user can override the starting point.

DESCRIPTION

This command invokes the IC Validator tool with an automatically gener- ated runset to calculate the metal density and density gradient values of the current design. Density calculated is for metal layers only.

EXAMPLES

The following example calculates the metal density and density gradient values for M1 and M3 layers.

```
prompt> set_app_options -name signoff.report_metal_density.run_dir \
  -value my_report_run
prompt> signoff_report_metal_density -select_layers {M1 M3}
```

The following example calculates the metal density and density gradient values for all regions in the rectilinear polygon {{0 0} {100 0} {100 100} {50 100} {50 50} {0 50}} .

```
prompt> set_app_options -name signoff.report_metal_density.min_density \
  -value {{M1 20} {M2 15}}
prompt> signoff_report_metal_density \
  -coordinates {{0 0} {100 0} {100 100} {50 100} {50 50} {0 50}}
```

The following example will generate the output density report as "filename.txt" and density gradient report as "filename.txt.gradient".

```
prompt> set_app_options \
  -name signoff.report_metal_density.density_window \
  -value {{M2 30} {M3 40}}
prompt> signoff_report_metal_density -output filename.txt
```

The following example will calculate the metal density and density gradient values from starting point (0, 5).

```
prompt> set_app_options \
  -name signoff.report_metal_density.density_window_step \
  -value {{M3 10} {M4 20}}
prompt> signoff_report_metal_density -starting_point {0 5}
```

SEE ALSO

set_host_options(2)
report_host_options(2)
remove_host_options(2)

sim_assertion_control

Control the behavior of assertions during low power verification.

SYNTAX

status **sim_assertion_control**
[-elements *element_list*]
[-exclude_elements *exclude_list*]
[-domain *domain_name*]
[-model *model_name*]
[-controlling_domain *domain* | -control_expr *boolean_expression*]
[-type *<reset | suspend | kill>*]
[-transitive *[<TRUE | FALSE>]*]

Data Types

element_list list
exclude_list list
model_name string
domain_name string
boolean_expression string

ARGUMENTS

-elements *element_list*

The list of instances or assertions.

-exclude_elements {*exclude_list*}

The list of design elements to which the command does not apply.

-domain *domain_name*

The domain in which to search for assrtions for this command.

-model *model_name*

The model in which to search for assertions.

-controlling_domain *domain_name*

The domain whose primary supply sets simstate controls when the assertion control is active. The DEFAULT controllling domain for each assertion is the domian in which that assertion is contained.

-type *<reset|suspend|kill>*

Specifies the type of assertion control. The default is reset.

-transitive *<TRUE | FALSE>*

Specifies if the command applies to the specified scope or the specified scope and all dependent subtrees.

If *-transitive* is not specified at all or if *-transitive* is specified without a value, the default is *-transitive TRUE*.

DESCRIPTION

The *sim_assertion_control* command provides a mechanism to disable assertions during low power verification. This enables the use of non-power-aware assertions in a power-aware environment. An example is assertions that check against X are typically not valid during power shutoff.

This command is for functional verification only does not have any impact on the implementation tools.

Implementation tool will not parse and check the options and it will not preserve this command in save_upf.

EXAMPLES

The following example selects all assertions in module modA.

```
prompt> sim_assertion_control -model modA  \
     -controlling_domain PD1
```

The following example selects all assertions labeled InstX/Assert1 in the module modA.

```
prompt> sim_assertion_control -model modA  \
     -elements InstX/Assert1
```

SEE ALSO

sim_corruption_control(2)

sim_corruption_control

Provides the ability to disable the corruptions of a specific set of design elements or types of design elements.

SYNTAX

status **sim_corruption_control**
-type <real | integer | seq | comb | port | process>
[-elements *element_list*]
[-exclude_elements *exclude_list*]
[-model *model_name*]
[-domain *domain_name*]
[-transitive *[<TRUE | FALSE>]*]

Data Types

element_list list
exclude_list list
model_name string
domain_name string

ARGUMENTS

-type <real | integer | seq | comb | port | process>

Specifies the type of design elements to which this command applies.

-elements *element_list*

The list of design elements to which the corruption control applies.

-exclude_elements {*exclude_list*}

The list of design elements to which the command does not apply.

-domain *domain_name*

The domain in which the command will apply.

-model *model_name*

The hdl model to which the command will apply.

-transitive *<TRUE | FALSE>*

Specifies if the command applies to the specified scope or the specified scope and all dependent subtrees.

If *-transitive* is not specified at all or if *-transitive* is specified without a value, the default is -transitive TRUE.

DESCRIPTION

This command turns off corruption semantics for specific set of design elements or type of design elements. The intent of this command is to allow behavioral modules to operate properly in a low power environment. For example, many real number modules require special corruption behavior to operate correctly.

This command is for functional verification only does not have any impact on the implementation tools.

Implementation tool will not parse and check the options and it will not preserve this command in save_upf.

EXAMPLES

The following example selects all real nets that start with L but excludes those that start with L1.

```
prompt> sim_corruption_control -type real \
     -elements [find_objects . -pattern "L*" -object_type net]
     -exclude_elements [find_objects . -pattern "L1*" -object_type net]
```

The following example selects integers in modules that belong to domain PD1.

```
prompt> sim_corruption_control -type integer \
     -domain PD1
```

SEE ALSO

set_simstate_behavior(2)

sim_replay_control

Specify initial blocks to be replayed when a domain powers up.

SYNTAX

status **sim_replay_control**
[-elements *element_list*]
[-exclude_elements *exclude_list*]
[-model *model_name*]
[-domain *domain_name*]
[-controlling_domain *domain*]
[-transitive *[<TRUE | FALSE>]*]

Data Types

element_list list
exclude_list list
model_name string
domain_name string

ARGUMENTS

-elements *element_list*

The list of intial blocks and hierarchical instances to which the command applies.

-exclude_elements {*exclude_list*}

The list of design elements to which the command does not apply.

-model *model_name*

The module in which to search for initial blocks.

-domain *domain_name*

The domain in which to search for initial blocks.

-controlling_domain *domain_name*

The domain whose primary supply set's simstate controls when the initial blocks is replayed. The DEFAULT controllling domain for each initial block is the domian in which that initial block is contained.

-transitive *<TRUE | FALSE>*

Specifies if the command applies to the specified scope or the specified scope and all dependent subtrees.

If *-transitive* is not specified at all or if *-transitive* is specified without a value, the default is *-transitive TRUE*.

DESCRIPTION

The *sim_replay_control* command specifies the action to be taken on the selected elements during simulation or formal verification when the simstate of the related supply set transitions from *CORRUPT* to *NORMAL* after time 0. The *sim_replay_control* allows specific initial blocks to be replayed when a domain powers up. Each initial block targeted by this command will be replayed when the primary supply set of the domain that contains the initial block transitions into the *NORMAL* state.

This command is for functional verification only does not have any impact on the implementation tools.

Implementation tool will not parse and check the options and it will not preserve this command in save_upf.

EXAMPLES

The following example selects all initial blocks whose label starts with L but excludes those blocks whose label starts with L1.

```
prompt> sim_replay_control \
     -elements [find_objects . -pattern "L*" -object_type process] \
     -exclude_elements [find_objects . -pattern "L1*" -object_type process]
```

The following example selects all initial blocks inside the modules that belongs to domain PD1.

```
prompt> sim_replay_control -domain PD1
```

SEE ALSO

sim_corruption_control(2)
sim_assertion_control(2)

size_cell

Rebinds leaf cells to a new library cell that has the required drive strength (or other properties).

SYNTAX

collection **size_cell**
cell_list
-lib_cell *lib_cell* | *lib_cell*
[-max_distance_to_spare_cell *length* | -not_spare_cell_aware]

Data Types

cell_list	list
lib_cell	collection
length	float

ARGUMENTS

cell_list

Specifies the leaf cell to be rebound. Each cell must be in scope (at or below the current instance).

-lib_cell *lib_cell*

Specifies the library cell objects to which the specified cell is to be bound. In this case, the object is either a named library cell or a library cell collection. This is a required opton. The library cells specified must be logical equivalent to the library cell which will be rebound.

lib_cell

This positional option is exactly equivalent to **-lib_cell**. And it is mutual exclusive with **-lib_cell**.

-max_distance_to_spare_cell *length*

Specifies the maximum distance between eco cell and spare cell in freeze-silicon mode. If option is not specified, command will use the default value (5x site heights). If there is no valid spare cell found within this distance, command will not do the cell sizing and report a error message to user. The resized eco cell will be placed to overlap the matched spare cell and a new attribute "fs_mapped_cell_name" will be used to record the mapping relationship in both eco cell and spare cell. The feature of spare cell aware only works in freeze-silicon mode currently. This option is mutually exclusive with **-not_spare_cell_aware**.

-not_spare_cell_aware

Disable the feature of spare cell aware in freeze-silicon mode. By default, this option is off. The feature of spare cell aware only works in freeze-silicon mode. This option is mutually exclusive with **-max_distance_to_spare_cell**.

DESCRIPTION

The **size_cell** command changes the drive strength (or other properties) of a leaf cell by rebinding it to a new library cell that has the required properties. Like all other netlist editing commands, for **size_cell** to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. **size_cell** returns the collection of cells sized if successful and a NULL string if unsuccessful. Each cell in *cell_list* must be in scope; that is, at or below the current instance.

The **size_cell** command also supports the spare cell aware feature in freeze-silicon mode.

The *lib_cell* that is being swapped in must conform to the following restrictions:

- *lib_cell* must be functionally compatible with the current library cell to which the cells in *cell_list* are bound.
- *lib_cell* cannot be the same as the current library cell of any of the cells in *cell_list*.
- *lib_cell* must have the same pin count and pin directions as the current library cell of the cells in *cell_list*. **size_cell** will match *lib_cell* by its pin numbers.

  The **size_cell** command is for leaf cells only. **size_cell** is optimized for incremental timing and does not cause a full timing update.

  This command support editing a verilog module in the module aspect, through **edit_module**.

  This command honors design.eco_freeze_silicon_mode. When this app option is true, cell to size will be marked as spare cell and renamed by adding _Spare post-fix, while a new cell of the specified lib cell will be created with attribute is_fs_eco_add being true. This new cell will inherit the name and connection of the cell to size.

Don't touch cells and don't use library cells

When app option *eco.size_cell.honor_dont_touch* is set to true, cells with dont_touch attribute are skipped. When app option *eco.size_cell.honor_dont_use* is set to true, **size_cell** checks if the new library cell has true value in dont_use attriute. **size_cell** errors out if the library cell has dont_use attribute set to true.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, an attempt to size a cell by a funtionally compatible library cell with the same pin count.

```
prompt> size_cell o_reg1 -lib_cell class/FD2P
Information: Sizing cell 'o_reg1' with lib cell 'class/FD2P'
{o_req1}
```

Editing a verilog module in the module aspect:

```
prompt> edit_module h1m {size_cell -lib_cell tcbn90ghvt/BUFFHVTD0 u1}
```

SEE ALSO

get_libs(2)
get_lib_cells(2)
report_cells(2)
set_reference(2)
edit_module(2)
eco.size_cell.honor_dont_touch(3)
eco.size_cell.honor_dont_use(3)

sizeof_collection

Returns the number of objects in a collection.

SYNTAX

int **sizeof_collection** [-categorize]
collection1

Data Types

collection1 collection

ARGUMENTS

collection1

Specifies the collection for which to get the number of objects. If the empty collection (empty string) is used for the *collection1* argument, the command returns 0.

-categorize

Return 0, 1, 2 indicating if the collection has 1, or or more than 1 item.

DESCRIPTION

The **sizeof_collection** command is an efficient mechanism for determining the number of objects in a collection.

EXAMPLES

The following example from PrimeTime shows a simple way to find out how many objects matched a particular pattern and filter in the **get_cells** command.

```
pt_shell> echo "Number of hierarchical cells: \
?           [sizeof_collection \
?             [filter_collection [get_cells * -hier \
?               "is_hierarchical == true"]]]"
Number of hierarchical cells is: 10
```

The following example from PrimeTime shows what happens when the argument for the **sizeof_collection** command results in an empty collection.

```
pt_shell> set s1 [get_cells *]
{"u1", "u2", "u3"}
pt_shell> set ssize [filter_collection $s1 "area < 0"]
pt_shell> echo "Cells with area < 0: [sizeof_collection $ssize]"
Cells with area < 0: 0
pt_shell> unset s1
```

SEE ALSO

collections(2)
filter_collection(2)

snap_cells_to_block_grid

Snaps the specified block cell instances to the block grid.

SYNTAX

status **snap_cells_to_block_grid**
-designs *design_list* |
-grid *grid* |
-cells *cell_list*

Data Types

design_list collection
grid block_grid_object
cell_list collection

ARGUMENTS

-designs *design_list*

Specifies the list of block reference designs whose instances are to be snapped to the associated block grid. This option is mutually exclusive with the **-grid** and **-cells** options, however, you must specify at least one of **-designs**, **-grid**, or **-cells**.

-grid *grid*

Specifies the block grid.

If this option is specified together with the **-cells** option, the user-specified cell instances are snapped to the specified block grid, regardless of which block grid the cells are associated with.

If this option is specified without **-cells**, all block cell instances whose reference design is associated with this block grid are snapped to the block grid. This option is mutually exclusive with the **-designs** option.

You must specify at least one of **-designs**, **-grid**, or **-cells**.

-cells *cell_list*

Specifies the cell instances to be snapped.

If this option is specified together with the **-grid** option, the cell instances are snapped to the specified block grid, regardless of which block grid the cells are actually associated with. If the specified block grid is not associated with a cell, the cell's origin is snapped to the grid. If the specified grid is the one that the cell is associated with, the snap-point setting on the cell's reference design would be honored.

If this option is specified without **-grid**, the cell instances are snapped to their associated block grids, respectively. And the snap-point setting on the cell's reference design would be honored.

You must specify at least one of **-designs**, **-grid**, or **-cells**.

DESCRIPTION

This command snaps the specified block cell instances to the block grid.

If you specify certain cell instances to snap to a block grid that are different from the block grid that the cell is associated with, you might see an error when running the *report_grids* command.

EXAMPLES

The following example finds all instances of the design BLK1 in the current design and snap them to the block grid associated with BLK1.

```
prompt> snap_cells_to_block_grid -designs BLK1
```

The following example uses the *grid* argument. The tool finds all the block reference designs associated with the specified block grid, then finds all the instantiations of the block references in the current design and snap them to the block grid.

```
prompt> snap_cells_to_block_grid -grid [get_grids gr1]
```

The following example uses the **-cells** argument. The tool snaps all the specified cell instances to their respective block grids if the cell has an associated block grid.

```
prompt> snap_cells_to_block_grid -cells [get_cells A1/U1]
```

The following example uses both the **-cells** and **-grid** options. The tool snaps all the specified cell instances' origin to the given block grid.

```
prompt> snap_cells_to_block_grid -cells [get_cells B1/M1] -grid gr1
```

SEE ALSO

create_grid(2)
get_grids(2)
remove_grids(2)
report_grids(2)
set_block_grid_references(2)
set_grid(2)

snap_object_shapes

Snap the edges if specified object to given grid. retained.

SYNTAX

status **snap_object_shapes**
-grid *grid*
[-left *left_mode*]
[-right *right_mode*]
[-bottom *bottom_mode*]
[-top *top_mode*]
[-edge_number *integer*]
[-edge_policy keep | increase | decrease | nearest]
[*object_list*]

Data Types

grid string
left_mode string
right_mode string
bottom__mode string
top_mode string
object_list collection

ARGUMENTS

object_list

Collection or selection set containing objects to snap. If this option is not specified, the global selection set is used.

-grid *grid*

Specifies the block or user grid to snap object edges.

-left *left_mode*

Specifies snapping policy for all left edges. Valid values are **keep**, **increase**, **decrease**, **nearest**, **keep_width**, **increase_width**, **decrease_width**, **nearest_width**. The X value of left edges could either be preserved, increase, decreased or rounded to nearest grid. Alternatively the X value of left edges might be set to preserve, increase, decrease or round the width to nearest grid.

-right *right_mode*

Specifies snapping policy for all right edges. Valid values are **keep**, **increase**, **decrease**, **nearest**, **keep_width**, **increase_width**, **decrease_width**, **nearest_width**. The X value of right edges could either be preserved, increase, decreased, rounded to nearest grid. Alternatively the X value of right edges might be set to preserve, increase, decrease or round the width to nearest grid.

-bottom ***bottom_mode***

Specifies snapping policy for all bottom edges. Valid values are **keep**, **increase**, **decrease**, **nearest**, **keep_width**, **increase_width**, **decrease_width**, **nearest_width**. The Y value of bottom edges could either be preserved, increase, decreased or rounded to nearest grid. Alternatively the Y value of bottom edges might be set to preserve, increase, decrease or round the height to nearest grid.

-top ***top_mode***

Specifies snapping policy for all top edges. Valid values are **keep**, **increase**, **decrease**, **nearest**, **keep_width**, **increase_width**, **decrease_width**, **nearest_width**. The Y value of top edges could either be preserved, increase, decreased, rounded to nearest grid. Alternatively the Y value of top edges might be set to preserve, increase, decrease or round the height to nearest grid.

-edge_number ***integer***

Specifies edge number to snap. The edge numbering starts with 1 (left edge) and going clockwise. The option is mutually exclusive with -left, -right, -bottom and -top.

-edge_policy keep | increase | decrease | nearest

Specifies snapping policy for given edge number.

DESCRIPTION

This command snaps object's sides to given grid by specified snapping policies.

Snapping is not performed if object is locked and global settings -ignore_locked is not enabled. Please see **set_edit_setting** for details.

EXAMPLES

The following example modifies currently selected object so that left/bottom edges are snapped to nearest grid and the width/height are preserved.

```
prompt> snap_object_shapes -grid my_grid -left nearest \
  -bottom nearest -right keep_width -top keep_height

prompt> snap_object_shapes -grid my_grid -edge_number 3 \
  -edge_policy nearest
```

The alternative syntax uses collection to specify objects.

```
prompt> snap_object_shapes [get_selection] -grid my_grid -left nearest
```

SEE ALSO

change_selection(2)
get_edit_setting(2)
get_selection(2)

set_edit_setting(2)

snap_objects

Snaps specified objects to the default snap type. An object is snapped onto the nearest position that satisfies the snapping constraint.

SYNTAX

status **snap_objects**
[*object_list*]

Data Types

object_list collection or selection

ARGUMENTS

object_list

Specifies the collection or selection set containing objects to snap. If this option is not specified, the global selection set is used.

DESCRIPTION

This command snaps the specified objects to the default snap types according to the object types of individual arguments.

The default snap type can be retrieved with the *get_snap_setting* command. Set the snap settings with the *set_snap_setting* command.

EXAMPLES

The following example snaps the cell "abc" to the nearest site row slot.

```
prompt> get_snap_setting -class std_cell
row_site
prompt> change_selection [get_cells "abc"]
prompt> snap_objects
```

The alternative syntax uses collection to specify objects.

```
prompt> snap_objects [get_cells "abc"]
```

SEE ALSO

get_selection(2)
change_selection(2)
move_objects(2)
rotate_objects(2)
copy_objects(2)
set_snap_setting(2)
get_snap_setting(2)
set_edit_setting(2)
get_edit_setting(2)

sort_collection

Sorts a collection based on one or more attributes, resulting in a new, sorted collection. The sort is ascending by default.

SYNTAX

collection **sort_collection**
  [-descending]
  [-dictionary]
  [-limit *value_count*]
  collection
  criteria

int *value_count*
string *collection*
list *criteria*

ARGUMENTS

-descending

Indicates that the collection is to be sorted in reverse order. By default, the sort proceeds in ascending order.

-dictionary

Sort strings dictionary order. For example "a30" would come after "a4".

-limit *value_count*

Only return the first unique values from the primary sort key.

collection

Specifies the collection to be sorted.

criteria

Specifies a list of one or more application or user-defined attributes to use as sort keys.

DESCRIPTION

You can use the **sort_collection** command to order the objects in a collection based on one or more attributes. For example, to get a collection of leaf cells increasing alphabetically, followed by hierarchical cells increasing alphabetically, sort the collection of cells using the *is_hierarchical* and *full_name* attributes as *criteria*.

The *criteria* can specify an attribute on another object. The other object must be available as an attribute on this object. For example, if you had a collection of net shapes, you can sort the objects by net using *owner_net.name* for instance.

In an ascending sort, Boolean attributes are sorted with those objects first that have the attribute set to *false*, followed by the objects that have the attribute set to *true*. In the case of a sparse attribute, objects that have the attribute come first, followed by the objects that do not have the attribute.

Sorts are ascending by default. You can specify ascending or descending order on a per attribute basis. You do this by adding a suffix to the attribute name in the sort criteria. The '-' means sort descending and '+' means sort ascending. If no suffix is specified than the default order ascending unless the *-descending* option is used.

The *-limit* option limits the size of the returned collection by choosing the first unique values from the primary sort criteria. For example using a limit of 1 will return all the objects with the smallest (or biggest if descending) value according to the primary sort key.

EXAMPLES

The following example from PrimeTime sorts a collection of cells based on hierarchy, and adds a second key to list them alphabetically. In this example, cells i1, i2 and i10 are hierarchical, and o1 and o2 are leaf cells. Because the *is_hierarchical* attribute is a Boolean attribute, those objects with the attribute set to *false* are listed first in the sorted collection.

```
pt_shell> set zc [get_cells {o2 i2 o1 i1 i10}]
{"o1" "i2" "o1" "i1" "i10"}
pt_shell> set zsort [sort_collection $zc {is_hierarchical full_name}]
{"o1" "o2","i1" "i10" "i2"}
pt_shell> set zsort [sort_collection -dictionary $zc {is_hierarchical full_name}]
{"o1" "o2" "i1" "i2", "i10"}
pt_shell> set zsort [sort_collection -dictionary $zc {area- full_name}]
{"i10" "o1" "o2" "i1" "i2" "i10"}
pt_shell> set zsort [sort_collection -dictionary $zc {area- full_name} -limit 1]
{"i10" "o1"}
```

SEE ALSO

collections(2)

source

Read a file and evaluate it as a Tcl script.

SYNTAX

string **source** [-echo] [-verbose] [-continue_on_error] *file*

string *file*

ARGUMENTS

-echo

Echoes each command as it is executed. Note that this option is a non-standard extension to Tcl.

-verbose

Displays the result of each command executed. Note that error messages are displayed regardless. Also note that this option is a non-standard extension to Tcl.

-continue_on_error

Don't stop script on errors. Similar to setting the shell variable sh_continue_on_error to true, but only applies to this particular script.

file

Script file to read.

DESCRIPTION

The **source** command takes the contents of the specified *file* and passes it to the command interpreter as a text script. The result of the source command is the result of the last command executed from the file. If an error occurs in evaluating the contents of the script, then the **source** command returns that error. If a return command is invoked from within the file, the remainder of the file is skipped and the source command returns normally with the result from the return command.

By default, source works quietly, like UNIX. It is possible to get various other intermediate information from the source command using the **-echo** and **-verbose** options. The **-echo** option echoes each command as it appears in the script. The **-verbose** option echoes the result of each command after execution.

NOTE: To emulate the behavior of the dc_shell **include** command, use both of these options.

The file name can be a fully expanded file name and can begin with a tilde. Under normal circumstances, the file is searched for

based only on what you typed. However, if the system variable sh_source_uses_search_path is set to "true", the file is searched for based on the path established with the search_path variable.

The **source** command supports several file formats. The *file* can be a simple ascii script file, an ascii script file compressed with gzip, or a Tcl bytecode script, created by TclPro Compiler. The **-echo** and **-verbose** options are ignored for gzip formatted files.

EXAMPLES

This example reads in a script of aliases:

```
prompt> source -echo aliases.tcl
alias q quit
alias hv {help -verbose}
alias include {source -echo -verbose}
prompt>
```

SEE ALSO

search_path(3)
sh_source_uses_search_path(3)
sh_continue_on_error(3)

split_clock_cells

Splits cells in the clock tree to fix DRC violations.

SYNTAX

status **split_clock_cells**
-cells *cell_list*
-loads *pin_port_list*

Data Types

cell_list collection
pin_port_list collection

ARGUMENTS

-cells *cell_list*

This option defines the list of cells that should get split in case their outputs on the clock network violate one of the DRC types. Either this option or the **-loads** option needs to be specified but they cannot be used together.

-loads *pin_port_list*

This option takes a list of pin or port collections and allows to define an explicit splitting of clock cells. All provided pins or ports in a collection need to have the same leaf driver. Each collection of pins or ports will get its own driver by splittingg the original driver. Either this option or the **-cells** option needs to be specified but they cannot be used together.

DESCRIPTION

This command can be used in two different modes. If the **-cells** option is specified then the command is used to split cells in the clock tree whose outputs violate DRC constraints. A typical application is to split integrated clock gates that drive a large amount of cells. Supported DRC constraints are: max transition, max capacitance, max net length, and max fanout. To fix max transition violations on the cell outputs and to avoid over-splitting of cells the input transition times of the cell are bounded by any given max transition constraint (coming either from the library or is set using the *set_max_transition* command).

The second mode of operation is enabled when the **-loads** option is given. Each collection of the given list of collections defines the fanout of a new or the original cell. If the common driving cell has a larger fanout than the specified collections, the remaining pins or ports of the fanout will become the fanout of one exclusive cell.

The cloned cells are named according to the following scheme: <old cell name>_split_<unique number>.

Splitting honors **set_dont_touch** constraints, **set_size_only** constraints, and fixed placement attributes on cells. Splitting is not

performed in case ports would have to be punched on power domain boundaries or frozen cells as defined by the **set_freeze_ports** command. Level-shifters and isolation cells are not split.

Splitting copies timing constraints, UPF constraints, user attributes, and clock balance points on cells as needed. For example, if a false path is defined through some of the split clock logic, the false path constraint is duplicated to all of the split cells as well. Net-based NDRs will get copied on the newly created output nets. The cloned cells are placed in the same logical hierarchy and voltage area as the original cell.

This command can be run before or after the *synthesize_clock_trees* command. In order to get a reasonable estimate of the fanout load of a cell all output nets of the specified cells are globally routed in case the nets have not been routed yet.

Additional verbosity output like assumed input transition times or detected constraints can be enabled by setting the app option *cts.common.verbose* to 1 or 5.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example splits two cells in case their outputs violate DRC constraints.

```
prompt> split_clock_cells -cells [get_cells {u1 u2}]
```

The following example explicitly splits the driver of the given collections of pins. Assume the output Y of cell c1 drives the load pins a1, a2, a3, and a4. Similarly, output Y of cell c2 drives the load pins b1, b2, b3, and b4. Then the following command execution results in two new clones of cell c1. One clone drives only a1, the second clone drives a2, and the original cell c1 drives a3 and a4. Similarly, it will result one new clone of cell c2. New clone drives b1 and b2, and the original cell c2 drives b3 and b4, for example:

```
prompt> set p1 [get_pins a1]
prompt> set p2 [get_pins a2]
prompt> set p3 [get_pins {a3 a4}]
prompt> set p4 [get_pins {b1 b2}]
prompt> set p5 [get_pins {b3 b4}]
prompt> split_clock_cells -loads [list $p1 $p2 $p3 $p4 $p5]
```

SEE ALSO

set_max_transition(2)
set_max_capacitance(2)
set_freeze_ports(2)
set_dont_touch(2)
set_size_only(2)
set_placement_status(2)
cts.common.max_fanout(3)
cts.common.max_net_length(3)
cts.common.verbose(3)

split_constraints

Partitions chip-level constraints into separate top-level and block-level constraints.

SYNTAX

int **split_constraints**
[-modes *mode_list*]
[-corners *corner_list*]
[-design_subblocks *block_module_names*]
[-hier_abstract_subblocks *block_module_names*]
[-output *dir_name*]
[-force]
[-nosplit]
[-compress gzip | none]
[-sdc_only | -upf_only]
[-internal_percent *percent_list*]
[-feedthrough_percent *percent_list*]
[-logic_depth]
[-ignore_repeaters]

Data Types

mode_list list
corner_list list
block_module_names list
dir_name string
percent_list\P *list*

ARGUMENTS

-modes *mode_list*

Specifies the modes to write. If this option is not specified, the command writes all modes.

-corners *corner_list*

Specifies the corners to write. If this option is not specified, the command writes all corners.

-design_subblocks *block_module_names*

Retain all the internal constraints of subblocks in the budget. By default, when generating split constraints, the tool assumes that any subblock of a block will be represented by a single-level abstract. So the budget will "cut out" constraints that are not visible on the boundary of the abstract.

You can use the **plan.budget.all_design_subblocks** application option to automatically set this option for all blocks.

-hier_abstract_subblocks *block_module_names*

Retain some extra internal constraints of subblocks in the budget. By default, when generating split constraints, the tool assumes that any subblock of a block will be represented by a single-level abstract. So the budget will "cut out" constraints that are not visible on the boundary of the abstract. With this option, additional block-internal constraints will be kept in places that are necessary for correctly timing hierarchical abstracts.

-output *dir_name*

Specifies the name of the directory in which to write the constraint files and scripts. If this option is not specified, the command writes the files to the *./split* directory.

-force

Overwrites the output directory if it exists. The output directory is specified by the **-output** option. If this option is not specified and the output directory already exists, the command issues an error message and exits.

-nosplit

Writes out long constraint command lines, even if the line length is greater than 80 columns. Use this option to simplify post-processing of the output constraint files or when comparing the constraint files by using the UNIX **diff** command. If this option is not specified, the command breaks long constraint command lines near column 80 and inserts a line continuation character (\).

-compress gzip | none

Specifies the type of compression to use on the output file. If this option is not specified, the command skips compression and writes plain text files.

-sdc_only

Splits only the SDC file. If this option is not specified, the command splits both the SDC and UPF files.

-upf_only

Splits only the UPF file. If this option is not specified, the command splits both the SDC and UPF files.

-internal_percent *percent_list*

Specifies the input and output delay percentage which will serve as the internal block delay percentage and used to calculate the delay for the boundary pins of the block.

-feedthrough_percent *percent_list*

Specifies the input and output delay percentage which will be used to calculate the delay value for feedthrough pins of the block. If this option is not specified the feedthrough pins will be budgetted at 50% of the available clock cycle.

-logic_depth

Specifies that the delay for the boundary pins of the block will be calculated according to logic depth.

-ignore_repeaters

Specifies that the delay for the boundary pins of the block will be calculated accordng to logic depth by considering odd or even number of buffers and inverters if any. This option should only be specified along with -logic_depth option.

DESCRIPTION

This command converts full-chip constraints into top-level and block-level constraints for a hierarchical design. Do the following steps to prepare your design before running the **split_constraints** command.

- Load the full chip netlist without abstracts
- For multivoltage designs, load the full chip UPF and run **commit_upf**
- Load the full-chip SDC
- Declare the blocks with **set_budget_options -add_blocks**
- Run **split_constraints**

For each block specified with **set_budget_options -add_blocks**, the **split_constraints** command creates a separate subdirectory under the directory specified by the **-output** option. The name of the subdirectory corresponds to the name of a block. Additionally, the command creates a directory that corresponds to the top-level design with the name *{design_name}*.

The **split_constraints** command writes constraint files in each block directory, similar to those generated by the **write_script** command. The *{design_name}* directory contains the constraints for the top-level, with the block constraints stripped out.

Inside the directory, mode-specific information is written to files named mode*_{mode_name}*.tcl, and corner-specific information is written to files named corner*_{corner_name}*.tcl. Mode- and corner-independent information is written to the design.tcl file. The top.tcl file sources all other constraint files. You can source the top.tcl file to load all necessary constraints for a block or top-level.

Memory Considerations

The **split_constraints** command requires the full netlist view of your chip. You cannot use abstracts to generate constraints with the **split_constraints** command. For large chips, the full netlist view can consume a significant amount of memory. The **split_constraints** command is specifically designed to use less memory than other commands that process design constraints. In particular, **split_constraints** processes constraints without calculating or propagating delay values and significantly reduces memory overhead.

To preserve the memory advantage of **split_constraints**, you should not trigger a timing update with the **report_timing**, **report_constraint**, or **report_qor** commands while the full-chip netlist is loaded. After the **split_constraints** command completes, you can control memory usage by using abstracts in your design.

Multiply Instantiated Blocks (MIB)

The **split_constraints** command has a limited capacity to handle multiply-instantiated blocks, and creates a single combined set of constraints for each block. Here is a summary of how conflicts are handled when the top-level constraints differ across instances:

- If constant values propagate from the top level to the boundary of the block, and the same constant feeds every block instance, then the constant is applied in the split constraints for the block. This process is repeated in the **write_budgets** command.
- If constant values propagate from the top level to the boundary of the block, but the same constant is not present for every block instance, then a warning is issued and the constant is not applied in the split constraints for the block.
- If timing exceptions such as **set_false_path** or **set_multicycle_path** straddle the boundary of a multiply-instantiated instance, the exceptions are pushed down using dedicated virtual clocks on the block ports. The virtual clocks keep the timing paths and corresponding constraints separate from each other when the block is timed. This process is repeated in the **write_budgets** command.
- All other block constraints, including the clocks and internal timing exceptions, will be pushed down to the block based on the first instance of the MIB. The other instances will be ignored. So you should be careful to constrain the "core" of all of your blocks in the same way.

Based on the restrictions above, it is required that the core of every instance of an MIB operate identically. The instances should be driven by the identical clocks in identical places. And internal constraints of the block should be identical. If they are not, the

split constraints will not faithfully reproduce the behavior of all of your MIB instances. Note that the top-level clocks which feed the blocks do not need to have identical names, only identical waveform. Later in the flow, the **set_block_to_top_map** command can be used to adapt to the mismatch of names.

There might be cases where you want instances of an MIB block to work in differing modes. For example, one MIB instance might be clocked with a fast clock and another with a slower clock. Or perhaps you want the instances to have different constant values to configure the block inputs. You can accomplish this by using the **set_block_to_top_map** command before running **split_constraints**. The syntax would look something like the following:

```
prompt> set_block_to_top_map -block inst1 -mode {fast_clock func}
prompt> set_block_to_top_map -block inst2 -mode {slow_clock func}
```

By default, **split_constraints** creates block constraints with a mode setting that matches the top level. But in this case, the block constraints will be generated with two modes (fast_clock and slow_clock). And the constraints for inst1 and inst2 will be kept completely separate in the block. No merging will occur.

Hierarchical Splitting

The **split_constraints** and **write_budgets** commands have advanced options that allow you control how the boundaries between a logical or physical hierarchy are handled for timing. This section first describes the default behavior of **split_constraints**, then discusses the variants of that behavior. For the purpose of discussion, assume a design with three block levels: top, mid, and bot. Everything except the top must be declared as budget blocks by using:

```
prompt> set_budget_options -add_blocks {mid mid/bot}
```

The **split_constraints** works on each block declared in the **set_budget_options** statement. Constraints are written to characterize the timing paths and clocks contained in the block, and constraints are written to characterize timing paths the enter or pass through any nested blocks.

Constraints are not written for timing paths that are fully contained in the "core" of nested blocks. For example, the constraints written for block "mid" will include everything that applies to mid itself *and* constraints for all timing paths that cross between mid and bot. But by default, constraints that apply to paths in the core of bot are not included in the split constraints for mid. Boundary constraints for the input and output ports for mid (for paths that cross from top to mid) are also *not* included by **split_constraints**. The boundary constraints come for paths at input and output ports of a block come from the **write_budgets** command.

There are two command line options of the **split_constraints** command that you might need to specify to modify which constraints are kept for a block. As the previous paragraph describes, constraints are not written for timing paths that are fully contained in the "core" of nested blocks. But you can override this default.

- You can use the *-design_subblocks* option to specify that nested modules in a design hierarchy will not be abstracted when they are used in their parent context. For example, if you specify "mid" as a design_sublock, then the split constraints written for "top" will also include *all* of the constraints for mid.
- You can use the *-hier_abstract_subblocks* option to specify that nested modules will be represented as a hierarchical abstract. When creating split constraints for a hierarchical abstract all constraints which apply to the boundary of that abstract will be included. Additionally constraints are also maintained along any timing paths that connect to hierarchically nested abstracts in the levels below. For example, if you specify "mid" as a hier_abstract_subblock, then the split constraints written for "top" will also include all of the boundary constraints for mid *and* all of the boundary constraints for bot.

In a multilevel hierarchy, the *-design_subblocks* and *-hier_abstract_subblocks* options may force constraints to cover many levels at one time. For example, if you are generating constraints for "top", and block "mid" is labeled as a design_subblock, then the constraints for top must include boundary constraints for "bot". In the following example, the generated constraints for the top level include all paths at the top, all paths in mid, and the boundary paths of bot.

```
prompt> split_constraints -design_subblocks mid
```

Splitting Clock Latencies

By default, the **split_constraints** command does not attempt to apportion clock latencies across blocks. Clock latencies are

pushed from the chip level down to each block, without change. For example, if the source clock latency of clock CLK1 is 2 at the top level, the source clock latency will still be 2 at the port boundary in the split constraints for each block.

You can use the **plan.budget.split_latencies** app option to cause the **split_constraints** command to reapportion and/or redefine clock latencies at block boundaries. This way you can adjust both the source latency and network latency seen during the synthesis of individual blocks.

Use the **-early_latency**, **-late_latency**, **-source**, **-early_dynamic**, and **-late_dynamic** options of **set_latency_budget_constraints** to control how latencies will be split.

For more details, see the **plan.budget.split_latencies** man page.

Multicorner-Multimode Support

By default, this command works on all scenarios. To specify different scenarios, use the **-corners** and **-modes** options.

EXAMPLES

The following example writes constraints for all blocks to the split_cons directory.

```
prompt> split_constraints -output split_cons
```

The following example selects values for splitting clock latencies to be used by split_constraints.

```
prompt> set_latency_budget_constraints -clock CLK1 \
        -early_latency 0.50 -late_latency 0.60 -default
prompt> set_latency_budget_constraints -clock CLK1:B1 -source \
        -early_latency 0.42 -late_latency 0.50 -default
prompt> set_app_options -name plan.budget.split_latencies -value prects
prompt> split_constraints
```

SEE ALSO

set_block_to_top_map(2)
write_script(2)
write_budgets(2)
plan.budget.all_design_subblocks(3)
set_latency_budget_constraints(2)
plan.budget.split_latencies(3)

split_fanout

Reduces the load on a net by inserting repeater cells. The command supports two modes: add a repeater cell on the subset of loads crossing different hierarchy, or insert a repeater cell to maintain the fanout count of driven loads is equal to or less than the maximum specified fanout.

SYNTAX

status **split_fanout**
[-driver *pin_port* | -net *net*]
[-hierarchy *hierarchy* | -on_route]
[-loads *pin_ports* | -max_fanout *number*]
-lib_cell *repeater_lib_cell*
[-respect_blockages]
[-max_distance_for_incomplete_route *length*]
[-net_prefix *prefix*]
[-cell_prefix *prefix*]

Data Types

pin_port	list
net	list
hierarchy	string
pin_ports	list
number	integer
repeater_lib_cell	string
length	float
prefix	string

ARGUMENTS

-driver *pin_port*

Specifies the driver on which to insert a repeater cell.

-net *net*

Specifies the net on which to insert a repeater cell. The **-net** option is mutually exclusive with the **-driver** option, only one of them can be specified.

-on_route

Enables on-route buffering by inserting buffers based on routing topology instead of logical connectivity. The **-hierarchy** option is mutually exclusive with the **-on_route** option, only one of them can be specified. Neither option is required. If the **-on_route** and **-loads** are used, all the loads must share the same net through the entire logical hierarchy and the buffer is added at the first common branch of routes. All the specified loads must have the same common branch of route data and no other loads can exist

under the same common branch. If **-on_route** is used, either **-driver** or -net can be used.

-loads ***pin_ports***

Specifies the loads on which to add repeater cells. The **-loads** option is required if the **-max_fanout** option is not used.

-max_fanout ***number***

Enables manual maximum fanout violation fixing based on input number. The number must be equal or greater than 2. If **-on_route** is used, the repeater is added based on the route data. Otherwise, the repeater is added based on the logical connectivity. The **-max_fanout** option is mutually exclusive with **-loads**, only one of them can be specified.

-max_distance_for_incomplete_route ***length***

Specifies a search distance for incomplete routes from the pin's terminal. This option is used to support the incompletely route nets. The default length value of **-max_distance_for_incomplete_route** is one repeater height. The net can be routed incompletely, but all route segments must be fully interconnected and only one wire within search distance can exist from each untouched load pin.

-hierarchy ***hierarchy***

Specifies the hierarchy to use when inserting repeater cells. This option can be used only when **-loads** is used. The **-hierarchy** option is mutually exclusive with **-on_route**, only one of them can be specified, but neither of them is required.

If **-hierarchy** is not specified, all specified loads must be connected to the same connection. Otherwise, the tool issues an error message.

-respect_blockages

Specifies that no repeater should be added inside a placement blockage, soft macro or hard macro. If the repeater is inside the blockage, the location to use when adding repeater cells will be moved to the new location toward the driver direction without overlapping the blockage. If the new location cannot be meet this requirement, an error message will be issued. You must specify the **-respect_blockages** together with the **-on_route** option.

-lib_cell ***repeater_lib_cell***

Specifies the library cell object to use as a buffer. If the specified library cell is inverting, an inverter pair is inserted instead of a buffer. Inverter pair cannot be used with the **-on_route** or **-max_fanout** option. This option is required.

-net_prefix ***prefix***

Specifies the prefix to use for new nets that are created when repeaters are inserted in the netlist.

By default, the command uses "eco_net" as the prefix.

-cell_prefix ***prefix***

Specifies the prefix to use for new repeater cells that are inserted in the netlist.

By default, the command uses "eco_cell" as the prefix.

DESCRIPTION

The **split_fanout** command has two features. The **-loads** option provides a way to add repeater on the subset of loads crossing different hierarchy. The **-max_fanout count** option inserts repeater cells to maintain the fanout count of driven loads is equal to or less than count. Port punching might be needed if the given loads are in a different logic hierarchy if **-loads** is used. If **-max_fanout** is used, the port punching might be needed if the physical topology and logical hierarchy are inconsistent.

If **-loads** is used without **-on_route**, the buffer is added at the center of the given loads. If both **-loads** and **-on_route** are specified, the buffer is added at the first common branch of routes.

If an inverting library cell is specified by **-lib_cell**, the **-on_route** and **-max_fanout** options cannot be used.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example inserts a buffer at the top level, and connects the driver dc1/ZN to the input pin of buffer. Port h3/enet and net h3/enet are created to connect the output pin of buffer to the loads {h3/lc1_2/I lc1_1/I}.

```
prompt> all_connected -leaf [get_nets -of dc1/ZN]
{dc1/ZN lc1_1/I h3/lc1_4/I h3/lc1_3/I h3/lc1_2/I}

prompt> split_fanout -driver dc1/ZN -loads {h3/lc1_2/I lc1_1/I} \
  -net_prefix enet -cell_prefix ecell
```

The following example inserts buffers using on-route buffering. In addition to the previous example, the buffer is added at the first branch of the existing route and the net data is split into two different nets.

```
prompt> split_fanout -on_route -loads {h3/lc1_2/I lc1_1/I} \
  -net_prefix enet -cell_prefix ecell -lib_cell lib/BUF
```

The following example inserts a buffer to ensure that max_fanout is equal to or less than 8. The grouping of loads is based on the order of the route data from loads to driver. Port punching might be required.

```
prompt> split_fanout -max_fanout 8 -net n7 -on_route \
  -net_prefix enet -cell_prefix ecell -lib_cell lib/BUF
```

The following example inserts a buffer to ensure that max_fanout is equal to or less than 8. Moreover, the buffer will be added in the location without overlapping any blockages.

```
prompt> split_fanout -respect_blockages -max_fanout 8 -net n7 -on_route \
  -net_prefix enet -cell_prefix ecell -lib_cell lib/BUF
```

SEE ALSO

all_connected(2)

split_macro_group

Split macro groups.

SYNTAX

collection_of_macro_groups **split_macro_group**
object object_list

Data Types

object collection of objects

ARGUMENTS

object collection_of_macro_group

Specifies macro groups to split.

DESCRIPTION

This command splits the macro group based on connectivity. It returns the collection of macro groups after splitting.

EXAMPLES

The following command splits macro group G1.

prompt> **split_macro_groups** G1

SEE ALSO

create_macro_group(2)

split_multibit

Splits a multibit cell into smaller multibit cells or single-bit cells.

SYNTAX

status **split_multibit**
cell_name
-lib_cells *list_of_lib_cells*
-path_groups *list_of_string*
-slack_threshold *float*
-cells *list_of_cells*
-exclude_instance *exclude_cells*
-verbose_file *file_name*

Data Types

cell_name string
list_of_lib_cells list
list_of_cells list or collection
exclude_cells list or collection

ARGUMENTS

cell_name

Specifies the name of the multibit cell that needs to be split. If neither "cell_name" or "-path_groups" is specified, tool will split all the multibit cells whose slack is less than threshold. If no option is specified, tool will split all the multibit cells with negative slack. This option is mutually exclusive with "-path_groups".

-lib_cells *list_of_lib_cells*

Specifies the list of smaller multibit cells or single-bit cells used to replace the multibit cell.

-slack_threshold *float*

Specifies the value of slack threshold. Tool will split multibit cells on the critical path whose slack is less than threshold. The default value of threshold is 0. This option is mutually exclusive with "cell_name" and "-lib_cells".

-path_groups *list_of_string*

Specifies the list of path_group names. Tool will split multibit cells belongs to path group list. If "-slack_threshold" is not specified, the multibit cells with negative slack will be split. If "-slack_threshold" is specified, the multibit cells with less slack than threshold will be split. This option is mutually exclusive with "cell_name" and "-lib_cells".

-cells *list_of_cells*

Specifies the list of multibit cells to be considered for splitting.

-exclude_instance *exclude_cells*

Specifies the list of multibit cells to be ignored for splitting.

-verbose_file *file_name*

Specifies the file name in which verbose messages and summary related to de-banking is dumped.

DESCRIPTION

The **split_multibit** command splits a multibit cell into smaller multibit cells or single-bit cells. The original cell is removed from the netlist and replaced by smaller multibit cells or single-bit cells.

The order of the library cells specified by using the **-lib_cells** option determines the pin connection order of the split cells.

The total bit-width of all the listed split library cells should equal the bit-width of the original multibit cell. If the total bit-width is too large, some bits of the split library cells are not used, then the pins of the unused bits are left dangling.

If a pin in the original multibit cell does not have a corresponding pin in the split library cell, the tool disconnects the pin and issues a warning message.

The new cells' name is derived from "single_bit_list" attribute of original multibit cell. If the 'single_bit_list' attribute is missing, the command will derive new cell's name based on original multibit cell's name and bit width.

EXAMPLES

The following example splits a 6-bit multibit cell into two smaller 3-bit cells:

```
prompt> split_multibit reg_6_bit_inst \
  -lib_cells { mylib/REG_3_BIT mylib/REG_3_BIT }
```

The following example splits a 2-bit multibit cell into two single-bit cells. If the 'single_bit_list' attribute of original multibit cell is "reg1 reg2", then the newly created single-bit cells are named by "reg1" and "reg2". If the 'single_bit_list' attribute is missing, then the new cells will be named as "reg_orig_inst_bank[0]" and "reg_inst_bank[1]"

```
prompt> split_multibit reg_orig_inst \
  -lib_cells { mylib/REG mylib/REG }
```

EXAMPLES

The following example splits multibit cells with negative slack and belongs to path group.

```
prompt> split_multibit -path_groups clk1_path
```

SEE ALSO

identify_multibit(2)
create_multibit(2)

split_objects

Splits a collection of net shapes and returns the collection of net shapes after the split.

SYNTAX

collection **split_objects**
[-line *line*]
[-rect *rectangle*]
[-ignore_end_cap]
[-gap *distance*]
[-gap_min_spacing]
[-force]
[-simple]
[*object_list*]

Data Types

object_list collection or selection
line string
rectangle string
distance float

ARGUMENTS

object_list

Collection or selection set containing objects to split. If this option is not specified, the global selection set is used.

-line *line*

Specifies the line to split objects by.

-rect *rectangle*

Specifies the rectangle to split objects by.

-ignore_end_cap

Ignores end caps when splitting wires.

-gap *distance*

Specifies the cut gap distance between resulting object edges.

-gap_min_spacing

Uses minimum spacing as the split gap.

-force

Resizes locked or fixed objects. By default, such objects are not resized according to global edit settings. Edit settings can be accessed using **get_edit_setting**.

-simple

Disable snapping and editing constraints. By default, objects are snapped according to global snap settings. Snap settings can be accessed using **get_snap_setting**.

DESCRIPTION

This command splits one or more specified objects skipping all fixed objects. The following objects are supported: movebound shapes, voltage area shapes, corridor shapes, net shapes, and pin shapes.

Snapping is performed only on the cut line or rectangle according to global snap settings. It is assumed that the original shapes are already correct.

The command returns a collection of the split objects if *object_list* is a collection of objects. Otherwise it splits objects in specified selection set and returns status.

EXAMPLES

The example splits the currently selected objects object along the line ((100,100) (200, 100)).

prompt> **split_objects -line {{100 100} {200 100}}**

The alternative syntax uses collection to specify the objects to split.

prompt> **split_objects [get_selection] -line {{100 100} {200 100}}**

SEE ALSO

change_selection(2)
create_shape(2)
get_selection(2)
get_shapes(2)
get_snap_setting(2)
merge_objects(2)
set_snap_setting(2)

split_polygons

Decomposes geometric objects into trapezoids, creating a collection of geo_mask or poly_rect objects.

SYNTAX

collection **split_polygons**
[-objects *object_list*]
[*pos_object_list*]
[-output poly_rect | geo_mask]
[-split horizontal | vertical]

Data Types

object_list collection
pos_object_list collection

ARGUMENTS

-objects *object_list*

Specifies the objects to be used to define the geometric region to be split. The object list can be a heterogeneous collection of poly_rect, geo_mask, shape, layer, and other physical objects.

If you specify a layer, the generated object includes the area of every shape in that layer. For other types of geometric objects, the generated object includes the area of each object in the list.

pos_object_list

Specifies the objects to be used to define the geometric region to be split. This positional option is provided for compatibility with other tools.

You must use either the **-objects** *object_list* option or the *pos_object_list* option; you cannot use both of these options together.

-output *poly_rect* | *geo_mask*

Specifies the type of object to return as a result, either **poly_rect** or **geo_mask**. The default is **geo_mask**.

-split *horizontal* | *vertical*

Specifies the direction of the fracture lines used for splitting the polygon, either **horizontal** or **vertical**. The default it horizontal.

DESCRIPTION

This command decomposes a specified region or list of regions into trapezoidal regions. If a fully trapezoidal decomposition is not possible, the result is a mixed set of trapezoidal and triangular regions. If the region is manhattan-angled (has no diagonal segments), the result is a set of rectangles.

By default, the command returns a geo_mask object. To return a poly_rect object instead, use the **-output poly_rect** option. A poly_rect object is a single geometric shape consisting of a set of coordinate points. A geo_mask object is a collection of poly_rect objects.

EXAMPLES

The following example returns a collection of geo_mask objects with trapezoidal regions, a decomposition of all the shapes on layer M1.

```
prompt> split_polygons -objects [get_layers M1]
```

SEE ALSO

copy_to_layer(2)
create_poly_rect(2)
create_geo_mask(2)
compute_polygons(2)
resize_polygons(2)
compute_area(2)
transform_polygons(2)

split_rdl_routes

Splits the redistribution layer (RDL) route into parallel routes and optionally copies the RDL route to the specified adjacent layer.

SYNTAX

status **split_rdl_routes**
[-nets *collection_of_nets* | -nets_in_file *nets_file*]
[-objects *collection_of_objects*]
[-mode same_layer | adjacent_layer]
[-widths {*layer_widths_list*}]
[-spacings {*layer_spacings_list*}]
[-number_of_routes {*layer_number_list*}]
[-from_layers {*layer_list*}
-to_layers {*layer_list*}
-via_interval *distance*]

Data Types

collection_of_nets collection
nets_file string
collection_of_objects collection
layer_widths_list list of string and float numbers pairs
layer_spacings_list list of string and float numbers pairs
layer_number_list list of string and integer pairs
layer_list list of string
distance float

ARGUMENTS

-nets *collection_of_nets*

Specifies the RDL nets to split.

-nets_in_file *nets_file*

Specifies the name of the file that contains the list of RDL nets to split.

-nets has higher priority than **-nets_in_file**.

-objects *collection_of_objects*

Specifies the RDL nets (see **-nets**) by cells or pins. Only the sub-nets which involve the cells or pins are affected.

If the app option **flip_chip.route.routing_style** is **spanning_tree**, the whole nets of the cells or pins are regarded as involved.

If neither **-nets**, **-nets_in_file**, nor **-objects** option is specified, all RDL nets are split.

-mode same_layer | adjacent_layer

Specifies the mode used to split RDL nets. Valid mode values are

- **same_layer** (the default)
  In this mode, the command splits the original RDL routes into several parallel, thinner routes on the same layer. You can use the **-widths**, **-spacings**, and **-number_of_routes** options to control how the routes are split. You can either split the original RDL route or copy it, but you cannot combine both operations. By default, **same_layer** mode is used to split the RDL nets.

- **adjacent_layer**
  In this mode, the command copies the RDL nets from the source layers specified by the **-from_layers** option to the target layers specified by the **-to_layers** option. If the copied routes create DRC violations with other shapes in the target layers, the tool automatically trims the violating piece of the route. If a blockage or metal shape exists on the destination layer, the tool creates a route around the blockage or metal shape. If the tool cannot create the shape, the tool issues an error message.

  After copying the routes, the command inserts vias between the source layers and the target layers along the target nets using the spacing specified by the **-via_interval** option.

  The **-from_layers**, **-to_layers**, and **-via_interval** options are required when you use the **adjacent_layer** mode. You can either split the original RDL route or copy it, but you cannot combine both operations.

-widths {*layer_widths_list*}

Specifies the width values after split for each given metal layer. The list format is:

{*layer_name1* {*width11 width12 width13* ...}
layer_name2 {*width21 width22 width23* ...}
layer_name3 {*width31 width32 width33* ...}
...}

The layer name is as specified in the technology file. The tool splits the routes into thinner routes based on the ordering of the width values. If only one width value is given, the tool splits the routes into thinner routes of equal widths. The units are microns.

The tool must maintain at least minimum spacing between each thin route. For each layer, the width of the original route shape should be greater than or equal to the summation of widths + minSpacing * (number_of_route-1).

If this option is omitted, the tool automatically calculates the width by dividing the original route width by the value specified by **-number_of_routes**. If the **-number_of_routes** option is omitted, the tool adjusts the value so that the split width is less than or equal to the maxWidth of the layer and the number of routes is at least two. The tool maintains at least minSpacing in between the split routes.

This option is valid only for the *same_layer* mode.

-spacings {*layer_spacings_list*}

Specifies the spacing values between the routes after split for each given metal layer. The list format is:

{*layer_name1* {*spacing11 spacing12 spacing13* ...}
layer_name2 {*spacing21 spacing22 spacing23* ...}
layer_name3 {*spacing31 spacing32 spacing33* ...}
...}

The layer name is as specified in the technology file. The tool maintains spacing between the split routes based on the ordering of the spacing values. If only one spacing value is given, the tool maintains equal spacing between according to the spacing value. The units are microns.

If the summation of the widths + summation of the spacing does not equal the original width, the tool does not perform the split operation.

If this option is omitted, tool automatically maintains equal spacing between the routes. The tool must maintain at least minimum spacing between each thin route. For each layer, the width of the original route shape should be larger or equal to summation of widths + minSpacing * (number_of_route-1).

This option is valid only for the *same_layer* mode.

-number_of_routes {*layer_number_list*}

Specifies the number of routes after split for each given metal layer. The list format is:

{*layer_name1 number1*
layer_name2 number2
layer_name3 number3 ...}

You must specify the layer name by using the layer name from the technology file.

If either the **-widths** or **-spacings** options are given in a list form, the value of **-number_of_routes** should be equal to the length of the **-widths** list or the length of **-spacings** list + 1.

If this option is omitted, the tool automatically calculates the number of routes by dividing the original route width by the **-widths** value. If the **-width** option is omitted, the tool adjusts the width value so that the split width is less than or equal to the maxWidth and the number of routes is at least two. The tool adjusts the value so that at least minSpacing is maintained between the split routes.

This option is valid only for *same_layer* mode.

-from_layers {*layer_list*}

Specifies the source layer names from which you want to split the RDL nets. The routed pattern for the target nets is copied from these layers. The number of layers specified in the **-from_layers** option should match the number of layers specified in the **-to_layers** option. The layers specified by the **-to_layers** and **-from_layers** options should be adjacent.

This option is valid for *adjacent_layer* mode only.

-to_layers {*layer_list*}

Specifies the destination layer names to which you want to split the RDL nets. The routed pattern for the target nets is copied to these layers. The number of layers specified in the **-to_layers** option should match the number of layers specified in the **-from_layers** option. The layers specified by the **-to_layers** and **-from_layers** options should be adjacent.

This option is valid for *adjacent_layer* mode only.

-via_interval *distance*

Specifies the interval distance to use when inserting vias along the target nets. The interval should be no smaller than via enclosure size. If the interval distance is too small, the tool skips the crowded via and inserts via until next valid location. The tool inserts vias between the layers specified by the **-from_layers** and **-to_layers** options.

This option is valid only for *adjacent_layer* mode.

DESCRIPTION

This command splits the route shapes of routed RDL nets into parallel routes.

In *same_layer* (default) mode, the routes are split into thinner routes according to the specified width, spacing and number of routes. The thinner routes are on the same layer as the original route.

In *adjacent_layer* mode, the routes are copied from the original layer to an adjacent target layer. The tool inserts vias between the layers.

RDL nets are nets in flip-chip or 3DIC designs which are routed on the redistribution layer. In a flip-chip design, RDL nets connect a bump cell to driver I/O cells. In a 3DIC or interposer design, RDL nets can connect the following elements:

- A micro-bump cell and a micro-bump cell
- A micro-bump cell and a through silicon via (TSV) cell
- A micro-bump cell and driver I/O cell

Micro-bump cells are also called flip-chip pads.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example routes "NET1" with the RDL router using the given NDR rule. The tool splits the route into 3 thin routes with a width of 3 on the M8 layer and 2 thin routes with a width of 5 on the M9 layer.

```
prompt> create_routing_rule rdlRule \
      -spacings {M8 5 M9 5} -widths {M8 12 M9 12}
prompt> set_routing_rule NET1 -rule rdlRule
prompt> route_rdl_flip_chip -nets NET1 -layers {M8 M9}
prompt> split_rdl_routes -nets NET1 -widths {M8 3 M9 5} -number_of_routes {M8 3 M9 2}
```

The following example splits the route into 5 thin routes with widths of 4, 3, 4, 3, and 4 and a spacing of 1 between them. The original RDL net was routed with an NDR width of 22 on the M9 layer.

```
prompt> split_rdl_routes -widths {M9 {4 3 4 3 4}} \
  -spacings {M9 {1 1 1 1}} -number_of_routes {M9 5}
```

The following example splits the route into 2 thin routes with the width of 10. The original RDL net was routed with NDR width of 22 on M9 layer, a maxWidth of 10 and a minSpacing of 0.4.

```
prompt> split_rdl_routes
```

The following example routes a net with the RDL router, splits the route, and move the routes to lower metal layers.

```
prompt> route_rdl_flip_chip -layers {M9 M8}
prompt> split_rdl_routes -mode adjacent_layer \
  -from_layers {M9 M8} -to_layers {M8 M7} -via_interval 10
```

SEE ALSO

route_rdl_flip_chip(2)

spread_objects

Spreads specified objects between anchor objects or the specified objects parent.

SYNTAX

status **spread_objects**
[*object_list*]
[-anchor *object_list*]
[-parent]
[-from {*x y*}]
[-to {*x y*}]
[-margin]
[-vertical]

Data Types

object_list collection or selection
x float
y float

ARGUMENTS

object_list

Collection or selection set containing objects to spread. If this option is not specified, global selection set is used.

-anchor *object_list*

Specifies the objects which should target objects are spread between.

-parent

Specifies that objects are spread within parent bounding rectangle.

For terminals and I/O pads the parent object will be the die. For soft macro pins the parent object will be the soft macro. For all other objects the parent will be the core.

-from *value_point_rect*

Specifies that the objects are spread between two positions. The first position is specified by this argument and the second position is specified by the **-to** argument.

-to *value_point_rect*

Specifies that the objects are spread between two positions. The first position is specified by the **-from** argument and the second position is specified by this argument.

-margin

Uses extended objects boundaries. The block boundaries are extended with keepout margins. The voltage areas boundaries are extended with guard bands. To exclude certain block keepout margins from consideration, use the following command:

```
prompt> win_set_filter -class cell \
  -filter {hard_macro_margin hard_margin route_blockage_margin soft_margin}
```

-vertical

Spreads the specified objects vertically rather than horizontally when using the **-to**, **-from**, or **-parent** options.

DESCRIPTION

The command spreads the specified objects horizontally between the leftmost and rightmost anchors or fixed objects. If no fixed objects are specified, the leftmost and rightmost objects are used. To spread objects vertically, specify the **-vertical** option.

The anchor object or anchor position and to position mark a range over which the objects are spread. The objects are then evenly spaced (moved) over this range.

Note that snapping is done automatically global using the snap settings.

EXAMPLES

The following example moves the currently selected objects so that they are spread equally in the horizontal direction between parent edges.

```
prompt> spread_objects -parent
```

The alternative syntax uses a collection to specify objects.

```
prompt> spread_objects [get_selection] -parent
```

The following example moves the currently selected objects so that they are spread equally in the vertical direction between parent edges.

```
prompt> spread_objects -parent -vertical
```

SEE ALSO

align_objects(2)
change_selection(2)
distribute_objects(2)
get_edit_setting(2)
get_selection(2)
get_snap_setting(2)
set_edit_setting(2)
set_snap_setting(2)

snap_objects(2)

spread_spare_cells

Spreads spare cells evenly throughout a rectilinear region. The spare cells usually come through Verilog ECO or manual netlist editing commands.

SYNTAX

status **spread_spare_cells**
[-cells *cell_objects*]
| [-voltage_areas *voltage_area_list*]
[-boundary *{ {{llx lly} {urx ury}} | {{x1 y1} {x2 y2} ...} }*]
[-ignore_blockage_types *blockage_type_list*]
[-density_aware_ratio *percentage*]
[-random_distribution]

Data Types

cell_objects collection
voltage_area_list list
llx float
lly float
urx float
ury float
x1 float
y1 float
x2 float
y2 float
blockage_type_list list
percentage integer

ARGUMENTS

-cells *cell_objects*

Specifies a collection of spare cells to spread. The cells in the collection must be marked as spare cells. The command issues an error if any specified cell is not a spare cell. The option is mutually exclusive with **-voltage_areas**. By default, the command spreads all spare cells in the design.

-voltage_areas *voltage_area_list*

Specifies a list of voltage areas. The command spreads the spare cells that belong to specified voltage areas. This option is mutually exclusive with **-cells**. By default, the command spreads all spare cells in the design.

-boundary *{ {{llx lly} {urx ury}} | {{x1 y1} {x2 y2} ...} }*

Specifies the rectangular or rectilinear region in which to spread spare cells. The command does voltage area-aware spreading

in the user-specified region. A rectangle is specified by its lower-left and upper-right coordinates, that is, {llx lly} and {urx ury}. A polygon is specified by its points, that is, {x1 y1} {x2 y2} {x3 y3}, and so on. The unit is micron. The command issues an error if any spare cell to be spread does not belong to any voltage area within the user specified region. By default, the command performs wire spreading in the core area.

-ignore_blockage_types ***blockage_type_list***

Specifies a list of blockage types. The command allows spare cells to be placed inside the blockage if blockage type is specified by the option. The command supports all blockage types same as create_placement_blockage, but does not honor any placement constraint that the blockage has. If the option is not used, the spare cells can not be placed inside any placement blockage.

-density_aware_ratio ***percentage***

Specifies the percentage of spare cells to be spread with density aware. This option allows user to control the percentage of density aware for spare cell spreading. The percentage means the percentage of spare cells will be spread considering density of existing cells, and the rest will be spread evenly. The two spreading ways are both blockage aware.

- The default percentage is 100, that means all spare cells are spread with considering cell density;
- 20 means 20% spare cells will be spread based on cell density and 80% spare cells will be spread evenly across the design;
- 0 means all spare cells are spread evenly without considering cell density.

-random_distribution

Spreads spare cells randomly. The *-random_distribution* and *-density_aware_ratio* options are mutually exclusive. When the option is used, the tool will ignore the current cell density and spare cells are spread with random distribution.

DESCRIPTION

This command spreads spare cells evenly in rectilinear regions. The command spreads all spare cells by default, or based on the spare cells specified by the **-cells** or **-voltage_areas** option. The command spreads spare cells in the core area by default, or in the user-specified rectilinear region specified by the **-boundary** option. The command is blockage-aware, and does not place spare cells with a blockage or macro. The command is voltage area-aware and spreads the spare cells within the associated voltage area for the cell. The command automatically derives the region in which to spread the cells from the voltage area boundaries for the specified spare cells. For voltage areas with multiple regions, such as disjoint voltage areas, the spare cells are placed according to the available area ratio and the reference name.

The command supports both legalized and non-legalized designs. If there is insufficient space to spread the spare cells that belong to the voltage area, the command issues a warning and skips cell spreading in this voltage area.

The placement of spare cells is not guaranteed to be legal. You should run the **place_eco_cells -cells -legalize_only** command or the **legalize_placement** command to generate a legal placement as needed.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example spreads all spare cells within its associated voltage area.

```
prompt> spread_spare_cells
```

The following example spreads only the spare cells specified with the **-cells** option.

```
prompt> spread_spare_cells -cells [get_cells $spareCells]
```

The following example uses the **-voltage_areas** option to spread only the spare cells that belong to the VA1 and VA2 voltage areas.

```
prompt> spread_spare_cells -voltage_areas {VA1 VA2}
```

The following examples use the **-boundary** option to spread only the spare cells in specified rectilinear regions.

```
prompt> spread_spare_cells -boundary {{10.8 20.3} {40.4 50.1}} \
  -cells [get_cells $spareCells]

prompt> spread_spare_cells -boundary {{10.8 20.3} {40.4 50.1}} \
  -voltage_areas {VA1}

prompt> spread_spare_cells -boundary {{10.8 20.3} {40.4 20.3} {40.4 50.1} \
  {25.0 50.1} {25.0 30.3} {10.8 30.3} {10.8 20.3}} \
  -cells [get_cells $spareCells]

prompt> spread_spare_cells -boundary {{10.8 20.3} {40.4 20.3} {40.4 50.1} \
  {25.0 50.1} {25.0 30.3} {10.8 30.3} {10.8 20.3}}
```

The following example uses the **-ignore_blockage_types** option to allow spare cells to be placed inside the blockage with user-specified type.

```
prompt> spread_spare_cells -ignore_blockage_types {soft partial}
```

The following example uses the **-density_aware_ratio** option to allow user to control the percentage of spare cells to be spread with density aware.

```
prompt> spread_spare_cells -density_aware_ratio 100
prompt> spread_spare_cells -density_aware_ratio 80
```

The following example uses the **-random_distribution** option.

```
prompt> spread_spare_cells -random_distribution
```

SEE ALSO

add_spare_cells(2)
get_attribute(2)
get_voltage_areas(2)
report_voltage_areas(2)
create_placement_blockage(2)

spread_wires

Spreads the wires in the current design to improve the total short critical area.

SYNTAX

status **spread_wires**
[-min_jog_length *min_ratio*]
[-min_jog_spacing_by_layer_name {*list_of_layer_value_pairs*}]
[-pitch *number_of_pitches*]
[-timing_preserve_nets {*collection_of_nets*}]
[-timing_preserve_setup_slack_threshold *slack_value*]
[-timing_preserve_hold_slack_threshold *slack_value*]

Data Types

count integer
min_ratio integer
number_of_pitches float
collection_of_nets collection
slack_value float
list_of_layer_value_pairs collection

ARGUMENTS

-min_jog_spacing_by_layer_name *list_of_layer_value_pairs*

Sets the minimum jog spacing value for each layer. You must specify the layers by using the layer names in the technology file.

Use the following format to specify the minimum jog spacing for each layer:

{*layerName value*}

For example, to specify a minimum jog spacing of 0.07 for metal2 and 0.08 for metal3, specify {{metal2 0.07} {metal3 0.08}}.

By default, this value is set to the minimum spacing for the layer plus a half layer pitch.

-min_jog_length

Sets the minimum value of the ratio of jog length to layer pitch.

By default, the *min_ratio* value is 2.

-pitch *num_of_pitches*

Sets the spread value in terms of number of pitches.

By default, the *num_of_pitches* value is 0.5.

-timing_preserve_nets {*collection_of_nets*}

Specifies the nets to preserve timing on.

If you specify this option, the command considers the specified nets as timing critical and skips wire spreading on these nets and the adjacent nets on the same layer within two routing pitches.

-timing_preserve_setup_slack_threshold *slack_value*

Specifies the setup slack threshold for timing critical nets.

If you specify this option, the command skips wire spreading on all nets with a worst setup slack less than or equal to this value and the adjacent nets on the same layer within two routing pitches.

-timing_preserve_hold_slack_threshold *slack_value*

Specifies the hold slack threshold for timing critical nets.

If you specify this option, the command skips wire spreading on all nets with a worst hold slack less than or equal to this value and the adjacent nets on the same layer within two routing pitches.

DESCRIPTION

This command spreads the wires in the current design to reduce the total short critical area. When spreading, a piece of the original wire is broken and pushed away and jog wires are created at both ends to maintain a connection.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following example spreads the wires 0.5 pitches.

prompt> **spread_wires -pitch 0.5**

SEE ALSO

route_detail(2)
widen_wires(2)

start_auto_save

Starts auto-saving in the current session.

SYNTAX

status **start_auto_save**
-frequency *time_interval*
[-location *disk_location*]
[-allowed_disk_space *space_limit*]
[-environment_settings include | exclude]
[-action_on_exit clear | keep]

Data Types

time_interval	integer
disk_location	string
space_limit	integer

ARGUMENTS

-frequency *time_interval*

Specifies the minimum time interval(in seconds) after which auto-save is triggered. The frequency specified must be greater than 1800 seconds. If a frequency is not specified, the default value of 1800 seconds will be used.

-location *disk_location*

Specifies the location where the auto-save data is to be stored. By default, disk location is current working directory.

-allowed_disk_space *space_limit*

Specifies the amount of disk space(in gigabytes or megabytes) that can be consumed by auto-save data. Specify disk space with the following tags: G, g, GB, Gb, gb, M, m, MB, Mb or mb. When this option is not specified, there is no limit on the space used by the auto-saved data.

-environment_settings *include* | *exclude*

Specifies whether to include or exclude the session-specific app options and Tcl variables when auto-save is triggered.

By default, session environment settings are included in the auto-saved library.

-action_on_exit *clear* | *keep*

Specifies whether to clear or keep auto_saved data when you exit the tool.

By default, auto-saved data will be kept.

DESCRIPTION

This command controls the auto-saving of the library data in the current session. Auto-saving must be initialized by specifying its frequency and can be stopped later as required by using the stop_auto_save command.

Frequency, location, allowed_disk_space, action_on_user_tool_exit and environment_settings can be configured when initialized or resumed.

Auto-saving in the current session is not enabled by default.

EXAMPLES

The following example starts auto-saving library data regularly after an interval of 2000 seconds.

```
prompt> start_auto_save -frequency 2000
Information: Auto-saving started for the current session. (NDMUI-877)
```

The following example starts auto-saving library data regularly after an interval of 1800 seconds at location /u/temp

```
prompt> start_auto_save -frequency 1800 -location /u/temp
Information: Auto-saving started for the current session. (NDMUI-877)
```

SEE ALSO

stop_auto_save(2)
recover_auto_save(2)
remove_auto_save(2)
report_auto_save(2)

start_busplan_gui

Start the busplan register planning GUI dialog.

SYNTAX

int **start_busplan_gui**

DESCRIPTION

Starts the busplan register planning GUI dialog. If the dialog is already open, it will first be closed and then re-started. If the main GUI has not already been started, it will be started first.

EXAMPLES

prompt> **start_busplan_gui**

SEE ALSO

create_busplans(2)
gui_start(2)
start_gui(2)

start_gui

Starts the application GUI.

SYNTAX

string **start_gui**
[-file *script*]
[-no_windows]
[-offscreen 1|0]
[-- x_args ...]

string *script*

ARGUMENTS

-file "*script*"

The given script file is sourced before the GUI starts.

-no_windows

The GUI starts without showing the default window.

-offscreen

If the specified value is 1 then the GUI starts in offscreen mode.

If the specified value is 0 then the GUI starts in normal (X Display) mode.

-- x_args ...

X11-specific arguments to pass to the X connection.

DESCRIPTION

This command starts the application GUI from the shell prompt. It is ignored if the application GUI has already been started. This is an alias for the gui_start command.

Note the application can only ever connect to one X server during program execution, so changing the value of the DISPLAY environment variable after the first successful start_gui command has no effect.

EXAMPLES

The following example starts the application GUI.

shell> **start_gui**

SEE ALSO

stop_gui(2)
gui_start(2)
gui_stop(2)

stop_auto_save

Stops auto-saving in the current session.

SYNTAX

status **stop_auto_save**

DESCRIPTION

This command stops auto-saving library data in the current session. Auto-saving can be resumed later by using the start_auto_save command.

Auto-saving in the current session is not enabled by default.

EXAMPLES

The following example stops auto-saving of library data:

```
prompt> stop_auto_save
Information: Auto-saving stopped for the current session. (NDMUI-879)
```

SEE ALSO

start_auto_save(2)
recover_auto_save(2)
remove_auto_save(2)
report_auto_save(2)

stop_gui

Stops the application GUI.

SYNTAX

string **stop_gui**

ARGUMENTS

None.

DESCRIPTION

This command stops the application GUI and returns to the shell prompt. It is ignored if the application GUI has not been started or has been stopped. This is an alias for the gui_stop command.

EXAMPLES

The following example stops the application GUI and returns to the shell prompt.

shell> **stop_gui**

SEE ALSO

start_gui(2)
gui_start(2)
gui_stop(2)

suppress_message

Disables printing of one or more informational or warning messages.

SYNTAX

string **suppress_message** [*message_list*]

list *message_list*

ARGUMENTS

message_list

A list of messages to suppress.

DESCRIPTION

The **suppress_message** command provides a mechanism to disable the printing of messages. You can suppress only informational and warning messages. The result of **suppress_message** is always the empty string.

A given message can be suppressed more than once. So, a message must be unsuppressed (using **unsuppress_message**) as many times as it was suppressed in order for it to be enabled. The **print_suppressed_messages** command displays the currently suppressed messages.

EXAMPLES

When the argument to the **unalias** command does not match any existing aliases, the CMD-029 warning message displays. This example shows how to suppress the CMD-029 message:

```
prompt> unalias q*
Warning: no aliases matched 'q*' (CMD-029)
prompt> suppress_message CMD-029
prompt> unalias q*
prompt>
```

SEE ALSO

print_suppressed_messages(2)
unsuppress_message(2)
get_message_ids(2)
set_message_info(2)

swap_objects

Swaps two cells in the GUI.

SYNTAX

status **swap_objects**
[*object_list*]

Data Types

object_list collection

ARGUMENTS

object_list

Specifies a collection or selection set that contains the objects to swap. If this option is not specified, the global selection set is used.

DESCRIPTION

This command swaps two selected cells. This command only swaps two cells of same geometry. Upon completion of swap, the two cells should swap location as well as orientation.

You should use this command only for supported cell types such as standard cells, I/O pads, soft macros, hard macros, and so on. The swap action is not performed in the following scenarios.

- One or both cells are marked as fixed
- Width or height of the cells does not match

EXAMPLES

The following example swaps the two selected cells.

```
prompt> swap_objects
```

The alternative syntax uses a collection to specify the objects.

prompt> **swap_objects [get_selection]**

SEE ALSO

get_selection(2)
change_selection(2)

synthesize_clock_trees

Builds and optimizes clock trees based on the clock definition.

SYNTAX

status **synthesize_clock_trees**
[-clocks *clock_list*]
[-propagate_only]
[-postroute]
[-routed_clock_stage detail | detail_with_signal_routes]

Data Types

clock_list collection

ARGUMENTS

-clocks *clock_list*

Specifies the clocks to compile. Without this option the command synthesizes all currently defined clocks in all active modes.

-propagate_only

Propagates all clocks in all active scenarios. Ideal clock related settings such as ideal clock network latencies on pins are also purged. Clock uncertainty is not purged. This option should be used when more modes/scenarios are activated after CTS is done. It does not synthesize the clocks but only propagates the clocks.

-postroute

Runs post-route clock tree optimization. This option should be used after clock tree is synthesized and routed.

-routed_clock_stage detail | detail_with_signal_routes

Specifies the route stage of the design for post-route clock tree optimization. Possible values of this option are **detail** and **detail_with_signal_routes**. **detail** means only clock nets are detail routed; signal nets are not routed. **detail_with_signal_routes** means both clock nets and signal nets are detail routed. This option must be used together with **-postroute** option.

DESCRIPTION

This command synthesizes clock trees and updates the design database with the compiled clock trees. The compilation of the clock tree is skew driven. Optionally, this command can optimize compiled clock tree for slack metric.

The command will work on clocks in active scenarios. It will have clocks balanced if either setup or hold analysis is active. It will fix the transition violations on clock pins if max_transition analysis is active, and fix the capacitance violation on clock nets if max_capacitance analysis is active. At the end of this command it will invoke **mark_clock_trees** to mark clock synthesized attributes on clock objects.

Before running the **synthesize_clock_trees** command, the **set_lib_cell_purpose** command can be applied to select buffers or inverters which used during clock tree synthesis.

EXAMPLES

The following example shows that the lib_cells matching 'buf*' in the 'tech_lib' library should not be inserted for clock tree synthesis:

```
prompt> set_lib_cell_purpose -exclude cts {tech_lib/buf*}
```

The following example shows a command that performs clock tree synthesis for clocks 'CLK1' and 'CLK2':

```
prompt> synthesize_clock_trees -clocks {CLK1 CLK2}
```

The following example shows a command that reports the summary and timing characteristics of the compiled clock trees:

```
prompt> report_clock_qor -clocks {CLK1 CLK2}
```

The following example shows a command that performs clock tree synthesis for clocks from modes 'fun.mode' and 'bist.mode':

```
prompt> set cts_clocks [get_clocks -clocks {CK1 CK2} -mode fun.mode]
prompt> append_to_collection cts_clocks [get_clocks -clocks {BCLK1 BCLK2} -mode bist.mode]
prompt> synthesize_clock_trees -clocks cts_clocks
```

SEE ALSO

check_clock_trees(2)
mark_clock_trees(2)
report_clock_qor(2)
set_clock_tree_options(2)
set_lib_cell_purpose(2)
set_scenario_status(2)

synthesize_clock_trunk_endpoints

Run block level clock trunk planning (CTP).

SYNTAX

status **synthesize_clock_trunk_endpoints**
[-blocks *block_designs*]
[-clocks *clock_list*]
[-estimate_timing]
[-host_options host_option_name]
[-work_dir path]
[-auto_clock connected | local | all]

Data Types

block_designs list
clock_list collection

ARGUMENTS

-blocks *block_designs*

Specifies the list of child blocks for which clock trunk planning needs to be done. Block designs may reference to either design views or abstract views. If this option is not specified, then clock trunk planning is done for all child blocks at all hierarchy levels.

In a multi-level physical hierarchy design, clock trunk planning is done in bottom-up manner and the order is automatically inferred.

This option cannot over-ride the abstract generation behavior of the command, which is to generate abstracts - after doing clock trunk planning - only for the blocks referencing to abstract views.

-clocks *clock_list*

Specifies the clocks for which clock trunk planning is to be done. The *clock_list* can contain master clocks at the top-level as well as clocks at the block-level. The command creates a mapping between clock definitions at top-level and clock definitions at block level. The mapped clocks are then passed to option **"-clock"** of **synthesize_clock_trunks**, which is executed for individual blocks.

By default, all clocks are synthesized. All virtual clocks are skipped. So a block level clock whose master can not be traced to top-level clock is ignored.

-estimate_timing

This boolean option invokes VIPO on blocks referecing to abstract views - after block-level CTP is done. This will not invoke VIPO at top-level.

-host_options ***host_option_name***

Specifies that the given host option should be used for distributed processing. If not specified, the tool will look for the global host option and use the global host option if it has distributed processing settings.

Depending on the design hierarchy, the command might need to run clock trunk planning for more than one block. Some of the blocks can run at the same time, depending on the hierarchy nesting.

-work_dir ***path***

Specifies the directory to put all generated scripts and log files. The command needs to run clock trunk planning for one or more child blocks of the current block. Scripts and log files would be genereated for CTP and subsequent abstract view creation at the child blocks' level.

If this option is not specified, the *work_dir* setting for the specified host option would be used. If no **-host_options** is specified, the *work_dir* of the global host option would be used. If there is no global host option and no **-host_options** is specified, the default is *./work_dir*.

-auto_clock connected | **local** | **all**

Associates clocks at the block level with clocks at the top level. For example, your top-level clock might be called "SYS_CLK", and the same clock in the block might be called "CLK". You can set the association between block clock and top clock by using the **set_block_to_top_map** command, or by using **synthesize_clock_trunk_endpoints -auto_clock** and specifying the same rule. See the **set_block_to_top_map** man page for more details. Note that the **-clocks** and **-auto_clock** options are mutually exclusive.

You can select one of three rules for auto-clock mapping:

- **connected** - The "connected" rule associates block-level clocks with top-level clocks under the following conditions: If a block clock has its source at a block port (a boundary clock), it can be associated to a top-level clock that propagates to that port. If a block clock has its source on an internal pin sources (a local clock), the top-level clock source must be declared at the same pin.
- **local** - The "local" rule maps connected clocks. In addition, if a block clock has its source on an internal pin sources (a local clock) and has no associated top-level clock, a new top-level clock will be created with the same waveform as the block-level clock. The new top-level clock will be associated with the local block-level clock.
- **all** - The "all" rule maps clocks as in the "local" rule. In addition, if a block clock has its source at a block port (a boundary clock) and no top-level clock propagates to that port, a new top-level clock will be created with the same waveform as the block-level clock. The new top-level clock will be associated with the local block-level clock. The top-level boundary clock will allow timing analysis to be done, but it is not necessarily correct for signoff. You should consider removing or replacing this clock before signoff.

Note that if two block-level clocks are declared at the same boundary port, the "all" rule cannot resolve both of them. This is because it will not automatically declare two conflicting clocks in the top-level block. You must manually declare two top-level clocks, with the appropriate waveforms. After the top clocks exist, **-auto_clock** mapping can associate them with the block clocks.

DESCRIPTION

This command runs block-level clock trunk planning (CTP) for all or specified child blocks - based on option **"-blocks"**. A run script is generated for each child block. In a multi-level physical hierarchy design, the tool determines the nesting order of the blocks and run the block scripts in a bottom-up order.

The tool gets the constraint files from the constraint mapping file. See the man page for **set_constraint_mapping_file** for details of how to set up the constraint mapping file.

SDC constraints are loaded through command **source**. Operations **remove_modes -all** and **remove_corners -all** are performed before loading the SDC constraints for a block.

CTS_CONSTRAINT constraints are loaded through command **source**. Then **"synthesize_clock_trunks -create_anchors -clock <clock_list>"** is performed on the block. Block-level CTP will not be triggered for blocks for which CTS_CONSTRAINT file is not specified in set_constraint_mapping_file.

For the blocks referencing to abstract views, the command generates abstract views after running CTP. If the abstract view with timing information exists for a block, the same type of abstract is re-created. For others, abstract views are generated with timing_level set to "compact". Abstract views are not generated for blocks referencing design views.

The command automatically sources top level SDC constraints, after block-level clock trunk planning is done.

This command also saves the current block to disk.

EXAMPLES

The following example runs block-level CTP on all child blocks across all hierarchies.

```
prompt> synthesize_clock_trunk_endpoints
```

The following example runs block-level CTP on child blocks GPU0 and GPU1

```
prompt> synthesize_clock_trunk_endpoints -blocks {GPU0 GPU1}
```

The following example runs block-level CTP only for clock CLK in all the child blocks

```
prompt> synthesize_clock_trunk_endpoints -clocks {CLK}
```

SEE ALSO

synthesize_clock_trunks(2)
create_abstract(2)
load_block_constraints(2)
estimate_timing(2)

synthesize_clock_trunk_setup_hier_context

Prepare/Update hierarchical designs for clock trunk planning.

SYNTAX

status **synthesize_clock_trunk_setup_hier_context**
[-init]
[-commit]
[-host_option host_option_name]

Data Types

-init Boolean
-uninit Boolean
-host_option String

ARGUMENTS

-init

When this argument is specified, optimizable abstracts will be created for sub-blocks linked with abstracts. This argument is mutually exclusive with argument "-commit".

-commit

When this argument is specified, regular abstract will be created for sub-blocks linked with abstracts. This argument is mutually exclusive with argument "-init".

-host_option host_option_name

Specifies that the given host option should be used for distributed processing. If not specified, the tool will run in serial mode.

DESCRIPTION

This commans performs preparation (with argument "-init") or updation (with argument "-commit") for sub-blocks for clock trunk planning flow. With argument "-host_option", it can run the operations in distributed fashion.

EXAMPLES

The following example creates optimizable abstracts serially for sub-blocks linked with abstracts.

```
prompt> synthesize_clock_trunk_setup_hier_context -init
```

The following example creates regular abstracts for sub-blocks linked with abstracts. Here tool will distribute the jobs according to specified host options.

```
prompt> synthesize_clock_trunk_setup_hier_context -commit -host_option grdOptions
```

SEE ALSO

synthesize_clock_trunks(2)
set_host_options(2)

synthesize_clock_trunks

Performs automatic clock trunk planning for the specified clock or clocks.

SYNTAX

status **synthesize_clock_trunks**
[-clock *clocks*]
[-from *startStage*]
[-to *endStage*]
[-list_only]

Data Types

clocks collection of clocks
startStage string
endStage string

ARGUMENTS

-clock *clocks*

Specifies the clocks for which automatic planning is to be performed.

-list_only

Prints out the major stages that constitute the default flow for the **synthesize_clock_trunks** command in THO mode.

These stages are: **pin_constr_generation**, **read_pin_constr_and_place_pins**, **clock_trunk_synthesis**.

This option cannot be used with the **-from** or **-to** options.

-from *startStage*

Specifies the starting stage for the **synthesize_clock_trunks** command. Valid values are **pin_constr_generation**, **read_pin_constr_and_place_pins**, **clock_trunk_synthesis**.

The default is **pin_constr_generation**.

If you specify both the **-to** and **-from** options, the *startStage* stage cannot be preceded by the *endStage* stage in the order specified by the **-list_only** option.

-to *endStage*

Specifies the final stage for the **synthesize_clock_trunks** command. Valid values are **pin_constr_generation**, **read_pin_constr_and_place_pins**, **clock_trunk_synthesis**.

The default is **clock_trunk_synthesis**.

If you specify both the **-to** and **-from** options, the *startStage* stage cannot be preceded by the *endStage* stage in the order specified by the **-list_only** option.

DESCRIPTION

This command performs automatic clock trunk planning on the specified clocks, or on all clocks.

The following prerequisites must be completed before running the **synthesize_clock_trunks** command. This can be achieved by running **synthesize_clock_trunk_endpoints** command before **synthesize_clock_trunks**.

- Blocks must be shaped
- Block clock ports must be placed.
- Top-level flip-flops must be placed.
- Clock trunk endpoints must have been identified with **set_clock_trunk_endpoints** and appropriate phase delays must have been set.

The command performs following tasks:

- Uses CTS engine to synthesize physical clock tree topology at top level.
- Compensates for the phase delays in the block level clock trunk endpoints.
- Maximize the common clock path between blocks communicates with each other.

The command supports two flavors:

- Pushdown mode: Enabled when app-option plan.clock_trunk.flow_control is set to push_down before executing the command. In this mode, the tool will consider the blocks as read-only. Hence CTS will place the clock trunk cells on top of the block. Use **push_down_clock_trunks** command (after synthesize_clock_trunks) to commit the physical topology of the clock trunk. Options -from and -to of synthesize_clock_trunks are not applicable for CTP-Pushdown mode.

- THO mode: Enabled when app-option plan.clock_trunk.flow_control is set to optimize_subblocks before executing the command. In this mode the tool will consider the blocks as editable. pin_constr_generation stage generates topological pin constraints for clock-nets. read_pin_constr_and_place_pins stage read these pin-constraints and execute place_pins for all these synthesized clock nets.

In order to check if the design is ready for synthesize_clock_trunks command, please run the command check_design_for_clock_trunk_planning command.

EXAMPLES

The following example performs automatic clock trunk planning for the clock CLK in CTP-Pushdown mode.

```
prompt> set_app_options -name plan.clock_trunk.flow_control -value push_down
prompt> synthesize_clock_trunks -clock CLK
```

The following example performs automatic clock trunk planning in CTP-THO mode for all clocks in the current mode. Here, it only runs pin_constr_generation step

```
prompt> set_app_options -name plan.clock_trunk.flow_control -value optimize_subblocks
```

prompt> **synthesize_clock_trunks -clock [all_clocks] -from pin_constr_generation -to pin_constr_generation**

The following example performs automatic clock trunk planning for all clocks in all active modes.

prompt> **synthesize_clock_trunks**

SEE ALSO

create_clock_buffer(2)
check_design_for_clock_trunk_planning(2)
place_pins(2)
set_clock_trunk_endpoints(2)
push_down_clock_trunks(2)
synthesize_clock_trunk_endpoints(2)
plan.clock_trunk.flow_control(3)

synthesize_multisource_clock_subtrees

Synthesizes the subtrees in a structural multisource clock tree synthesis (CTS) flow.

SYNTAX

status **synthesize_multisource_clock_subtrees**
[-clocks *clock_list*]
[-list_only]
[-from preprocess | merge | optimize | route_clock | refine]
[-to preprocess | merge | optimize | route_clock | refine]

Data Types

clock_list collection

ARGUMENTS

-clocks *clock_list*

Specifies which clocks to synthesize. The subtree options are defined by the **set_multisource_clock_subtree_options** command. By default, all multisource clocks are synthesized.

-list_only

Lists the available steps for the subtree synthesis command. These steps are **preprocess**, **merge**, **optimize**, **route_clock**, and **refine**. By default, all five steps are run when the **synthesize_multisource_clock_subtrees** command is used. But, the *-from* and *-to* options can be used to limit the command to execute only the specified range of steps. *-list_only* simply lists the valid steps that can be passed to the *-from* and *-to* options.

-from preprocess | **merge** | **optimize** | **route_clock** | **refine**

Specifies that multisource subtree synthesis should start from the specified step. By default, all the steps are run in order: **preprocess**, **merge**, **optimize**, **route_clock**, and **refine**. This option can be used to start from a later step. For example, **synthesize_multisource_clock_subtrees -from optimize** skips the preprocess and merge steps and only execute the optimize, route_clock, and refine steps.

The *-from* and *-to* options can be used together or separately to run only certain steps of subtree synthesis, mainly for analysis purposes. Regardless of the usage of these options, all five steps should eventually be run, and run in the correct order.

-to preprocess | **merge** | **optimize** | **route_clock** | **refine**

Specifies that multisource subtree synthesis should end at the specified step. By default, all the steps are run in order: **preprocess**, **merge**, **optimize**, **route_clock**, and **refine**. This option can be used to end before completing all steps. For example, **synthesize_multisource_clock_subtrees -to optimize** runs the preprocess, merge, and optimize steps, but skips the route_clock and refine steps. The **-from** and **-to** options can be used together or alone to run only certain steps of subtree synthesis, mainly for analysis purposes. Regardless of the usage of these options, all five steps should eventually be run, and run

in the correct order.

DESCRIPTION

Structural multisource clock trees are clock trees where the user specifies the structure of the clock paths in the incoming netlist. The tool is not allowed to modify the sequence of clock cells types on any path from the multisource roots to the clock sinks. This means that structural multisource clock tree construction is restricted to merging, splitting (cloning), sizing and relocation of cells in the incoming netlist. No restructuring or (un)buffering of the clock nets is allowed, since this would modify the sequence of cell types on a path from multisource root to clock sink.

A multisource CTS tree has several components, including the pre-mesh tree driving the mesh, the mesh itself, the driver objects which are driven by the mesh, and the subtrees driven by those driver objects. The **synthesize_multisource_clock_subtrees** command is used to structurally synthesize the subtrees driven by the mesh.

Structural synthesis of multisource subtrees involves several steps, the end goal being a set of DRC-free subtrees that are built to meet your latency and skew targets.

Specifying Structural Subtree Synthesis Options and Constraints

Synthesis constraints should first be defined by using the **set_multisource_clock_subtree_options** command. This command performs four main functions:

- Defines driver objects for subtrees of a clock for balancing and sink assignment
- Defines which clock to balance in each subtree set
- Defines synthesis constraints like total maximum wire delays
- Enables level balancing and defines the target level number

A subtree set indicates that a set of clock fanouts and sinks are equivalent. Equivalent fanouts and sinks can be freely assigned to any driver object for the subtree set, and that they should be skew balanced together. This is done by specifying the driver objects that fan out to the sinks of that subtree set.

Each subtree set can be synthesized for only one clock. The clock needs to be specified with the **set_multisource_clock_subtree_options** command.

The synthesis style disallows buffer insertion or removal for opimization. Only the existing logic in the subtrees can be used, and that logic is sized, merged, split, and placed to meet the CTS goals. This structural approach requires a netlist that has been prepared specifically for this synthesis style, since no buffer insertion or removal can be done.

Buffers can only be inserted to balance the number of levels in the subtrees. This level balancing step is executed in the preprocess step and needs to be enabled by the **set_multisource_clock_subtree_options** command.

Default command tries to balance all the subtrees to same level. You can specify different target_level for the sinks using set_multisource_clock_subtree_constraints command.

Using structural mode is the only way to guarantee a certain number of clock levels to each sink. Since no buffer insertion or removal is performed (except for level balancing), the number of levels to each sink remain the same before and after subtree synthesis.

The netlist needs to have enough clock levels to allow sufficient splitting of logic to drive the clock trees. If not enough levels exist in the pre-CTS structural tree, this can create a very large pin load driven by the driver objects after subtree synthesis, since many split buffers or gates will have to connect to that driver stage.

You can also optionally enable ICG re-ordering feature with set_multisource_clock_subtree_options. ICG re-ordering is performed only if it saves power. You can prevent specific ICGs from re-ordering using set_multisource_clock_subtree_constraints command.

This is useful for designs where impact of dynamic power due to net switching activity is higher and clock gating is employed to address the same.

You can also optionally enable power aware subtree synthesis to allow optimization of subtrees for DRC and latency with focus to total clock tree power. This feature is false by default, and can be enabled using the app_option cts.multisource.subtree_synthesis_enable_power_optimization.

In addition to the constraints defined by **set_multisource_clock_subtree_options**, subtree synthesis also honors:

- **set_max_transition** DRC limits
- **set_max_capacitance** DRC limits
- Fanout limits set by the app option **cts.common.max_fanout**
- Latency targets set by **set_clock_tree_options -target_latency**
- Float and stop exceptions set by **set_clock_balance_points**
- Ignore exceptions set by **set_clock_balance_points -consider_for_balancing false**
- **set_placement_status**, **set_dont_touch**, and **set_size_only** constraints on cells in the clock tree
- Clock cell spacing rules set by **set_clock_cell_spacing**

Constraints that are not honored by subtree synthesis include:

- Skew targets set by **set_clock_tree_options -target_skew**
- Skew groups created by **create_clock_skew_group**
- Balance groups created by **create_clock_balance_group**
- Max net length limits set by the app option **cts.common.max_net_length**

Executing Structural Subtree Synthesis

Structural subtree synthesis is performed with the **synthesize_multisource_clock_subtrees** command. By default, this command operates on all clocks that have subtrees defined by the **set_multisource_clock_subtree_options** command.

Structural subtree synthesis consists of five main steps. First, level balancing is executed in the preprocess step, if enabled. Without a given target level, level balancing inserts buffers such that all sinks of an option set are at the same logic level in the clock tree. The command minimizes the number of inserted buffers by inserting them as early as possible in the subtrees. Constraints like don't-touches on nets or frozen hierarchy ports are honored during this process. If a target level is specified with the **set_multisource_clock_subtree_options** command, then buffers are inserted such that all sinks are at the given target level. Clock paths to sinks which are already at a higher level than the specified target, are not modified. The type of buffer will be selected from the available list of buffers enabled for CTS. Sizing and possibly splitting of these inserted buffers will happen at later optimization stages.

Next, all clock logic in the subtree group is merged as much as possible and assigned to a single driver object of that group. Merging all clock logic helps ensure that the subsequent splitting of clock tree logic is done optimally, based on the locations of the driver objects and the clock sinks.

Merging is done for all equivalent clock cells. The cells need to share common driver nets on each input pin to be considered equivalent for merging. Any timing constraints on the cells need to be the same for merging to occur.

Merging can occur across logical hierarchy boundaries, and merging honors any frozen ports defined by the **set_freeze_ports** command. **dont_touch** constraints on cells and nets in the clock tree can restrict merging. Any cells set as **size_only** or **fixed** are not be merged. Merging can only occur across cells that belong to the same library cells subset and that are within the same power domain, voltage area, and move bound.

Merging of certain cells can be disabled using the **-dont_merge_cells** option of the **set_multisource_clock_subtree_options**

command. Note, however, that cells specified with this option can still be split (see the next optimization step below). This means that a second execution of this command cannot merge these cells again unless the option set definition is altered and the cells are removed from the **-dont_merge_cells** list.

The naming convention for the merged cells can be configured through the following app options: cts.multisource.subtree_merge_cell_name_prefix, cts.multisource.subtree_merge_cell_name_suffix, and cts.multisource.subtree_merge_concatenate_length_threshold

After merging is complete, the next step in subtree synthesis occurs, which is splitting and optimization. As the clock trees are clustered and built, all clock logic is assigned to the appropriate subtrees based on nearest location, and splitting of clock logic is done as needed in order to divide and assign the sinks and clock fanouts to their subtrees. This optimization is intertwined with sizing and relocation of the clock logic.

Splitting and optimization honors frozen ports set with the **set_freeze_ports** command, **set_dont_touch** constraints, **set_size_only** constraints, and fixed placement attributes on cells. Splitting is not performed in case ports would have to be punched on power domain boundaries.

Splitting copies timing constraints, UPF constraints, and user attributes on cells as needed. For example, if a false path is defined through some of the split clock logic, the false path constraint is duplicated to all of the split cells as well. The cloned cells are placed in the same logical hierarchy and voltage area as the original cell.

After this splitting and optimization step, all logic is assigned to the appropriate subtrees, and the trees have been optimized to be DRC-free and to meet your latency targets.

The naming convention for the cells and nets created during splitting can be configured through the following app options:

cts.multisource.subtree_split_cell_name_prefix,
cts.multisource.subtree_split_cell_name_suffix,
cts.multisource.subtree_split_net_name_prefix, and
cts.multisource.subtree_split_net_name_suffix.

The next step in subtree synthesis is clock routing. All clock nets are routed in this stage. The routing approach is controlled by the **cts.multisource.subtree_routing_mode** app option, which can be set to **none** to do virtual routing of clock nets in the subtrees, **global** to do global routing of those nets, or **fishbone** to do fishbone routing. Fishbone routing creates a mix of detail routed fishbone trunks and fingers, and global comb routes to connect the net loads to the fishbone fingers.

The final step is incremental refinement. This step performs cell sizing and cell relocation based on global/fishbone routing. Relocation does not happen on cells connected to clock straps/fishbone routing. The refinement timing is based on extracted clock routes to have the most accurate timing and DRC picture available.

By default, **synthesize_multisource_clock_subtrees** runs these five steps when building subtrees. By using the **-from** and/or the **-to** options to this command, it can be restricted to only run certain steps. The valid step names are **preprocess**, **merge**, **optimize**, **route_clock**, and **refine**. Typically, these options are used to run only one component of the subtree synthesis and analyze results before continuing to the next step. In the end, all five steps should be run to fully synthesize the subtree.

Subtree synthesis generates logfile output, which gives basic guidance about what the command is doing and any problems encountered during execution. You can generate additional, verbose output by setting the **cts.multisource.verbose** application option to 1. This setting is intended for advanced users doing more detailed investigation of the multisource subtree synthesis results.

Driver Objects

In a structural multisource CTS flow, the subtrees can optionally connect to the mesh through a layer of cells i.e. driver objects. These cells are specified by the **set_multisource_clock_subtree_options -driver_objects** command and option. The driver object acts as the driver or root of a particular multisource CTS subtree. If the driver objects connect to subtrees through clock straps, the placement of first-level cells in the subtree is snapped to the driver object's routing geometries, to ensure efficient connection of the subtree to the clock straps.

Typically, a driver object is a clock buffer, but the multisource CTS flow allows a lot of flexibility in the driver object concept. The following are supported approaches for driver objects of subtrees:

(1) A standard cell, with its clock output pin defined as the driver object. The first-level cells in the subtree are snapped to the driver cell output pin geometry.

(2) The driver object is a standard cell output pin, which is connected to a net which has prerouted geometries. The first-level cells in the subtree are snapped to the prerouted geometries on the driver's net.

(3) There is nodriver cell, instead, the clock port terminals are used as driver objects. For example, there are eight driver objects associated with eight clock ports, clk[0:7]. Each port has different terminal shapes. The first-level cells in the subtree are snapped to the terminals.

(4) There is no driver cell, instead, the driver objects are clock ports that have prerouted net geometries. This differs from the previous example where each port had just the terminal shapes. In this case, there are also prerouted nets connecting to those terminals. The first-level cells in the subtree are snapped to the prerouted nets attached to the port terminals.

Based on the previous four examples, a driver object can be either a physical cell output or an input port of the block. If no prerouted net geometries exist, then the cell output pin or the port terminal shapes are used for snapping the first-level cells in the subtrees. If instead there are net shapes with the *shape_use==stripe* or *shape_use==user_route* properties set, those net shapes are used as snapping points for the first-level cells in the subtrees.

If a driver object net has preroute shapes with both *shape_use==stripe* and *shape_use==user_route* properties, the *stripes* setting take precedence over the *user_routes* setting and only those stripe shapes are used for snapping and routing to the first level in the subtrees. This distinction can be useful to define a preroute structure where the subtrees can only connect to certain shapes.

You can use the app option **cts.multisource.ignore_drc_on_subtree_driver** to specify that max_capacitance and/or max_anout constraints are to be ignored on the driver object pin or ports.

Latency and Skew Targets

The skew balancing behavior of multisource subtree synthesis is very similar to that of standard CTS. Subtree synthesis honors latency targets as specified by the **set_clock_tree_options -target_latency** command. If no target latency is set, then subtree synthesis works toward a minimum latency solution for the subtree group.

Subtree synthesis also honors stop, float, and ignore clock exceptions specified by the **set_clock_balance_points** and **set_clock_balance_points** commands. When using structural style subtree synthesis, no optimization is done beyond clock exceptions that are intermediate in the clock tree. It is safer to only use exceptions on sinks when doing structural subtree synthesis.

The **set_clock_tree_options -target_skew** command is not currently honored by multisource subtree synthesis. Skew is always minimized towards zero clock skew in a subtree group.

Skew groups and balance groups as specified by the create_clock_skew_group and create_clock_balance_group commands are not honored by multisource subtree synthesis.

Structural multisource subtree synthesis supports one additional approach to latency targeting that is not available in regular CTS, called *latency mode*. This mode is activated by specifying **set_clock_tree_options -target_latency 0** on the clock object being balanced for the multisource subtree group. In latency mode, the latency target for every clock sink is explicitly specified as a **set_clock_balance_points** float constraint. This differs from ordinary treatment of float constraints, which are normally considered offsets rather than absolute targets.

For example, typically if a balance point is specified on a sink with downstream_latency 200ps using the **set_clock_balance_points** command, that is meant to model 200ps of downstream latency from the balance point, which tells CTS to tap that sink 200ps earlier in the clock tree. But in latency mode, that 200ps constraint is instead treated as an absolute latency target to be achieved at that sink.

This mode of latency targeting and skew balancing is typically achieved using the **derive_clock_balance_points -reference_latency 0.0** command. If this command is run before **synthesize_multisource_clock_subtrees**, then the ideal latency at every sink is converted to a **set_clock_balance_points** float pin constraint. Note that this value includes both source and network **set_clock_latency** constraints seen at each sink in ideal mode.

For example, if sink A has an ideal network latency of 180ps set by the set_clock_latency command, this is translated to a **set_clock_balance_points -consider_for_balancing true -delay -0.180** constraint on that sink. Note that the polarity of the

constraint is reversed, as a negative balance point downstream_latency indicates that a sink should be delayed in the clock network by that amount.

All of this enables a flow where you explicitly specify the desired latency to every sink in ideal mode using **set_clock_latency** constraints. Then, by converting those ideal latencies to balance point constraints, and specifying a target latency of 0 on the clock to enable *latency mode*, multisource subtree synthesis can build the trees while targeting the ideal latencies.

Given all this, you can use one of two use models:

The standard approach, where a target latency is set with **set_clock_tree_options**, otherwise minimum latency clock trees are targeted. Optionally, balance point and ignore point constraints can also be applied to affect the skew balancing, just as with standard CTS.

The alternate *latency mode* approach, where achievable ideal mode latencies are specified for every sink by using **set_clock_latency** constraints. By setting a target latency of 0 and using the **derive_clock_balance_points -reference_latency 0.0** command, these latencies are converted to balance point constraints and targeted as absolute latency goals at each sink.

Fishbone Routing

Fishbone routing is one of the available routing approaches for subtree synthesis, configured by setting the **cts.multisource.subtree_routing_mode** app option to *fishbone*. Fishbone routing is a structured routing topology where the net driver feeds a central fishbone trunk that spans the cluster of net loads being driven. Fishbone routing is only done for a net if the trunk is at least five gcells in length.

Several fishbone fingers can extend from the central trunk. The net loads connect directly to the fishbone fingers by way of comb routes, typically one comb route for each net load. The frequency of these fingers is controlled by the **cts.routing.fishbone_max_tie_distance** app option, which specifies a maximum distance from a net load to its nearest fishbone finger.

Optionally, you can use an additional level of fishbone routing, called sub-fingers. The sub-fingers connect to the fingers, and then the comb routes can instead connect the net loads to the sub-fingers. This can reduce wire length, especially in the case that a large **fishbone_max_tie_distance** is used. The behavior of sub-fingers is controlled by the **cts.routing.fishbone_max_sub_tie_distance** app option.

After the subtree synthesis command completes, the trunk, fingers and sub-fingers are detail routed, and defined as preroutes, using the *shape_use==stripe* attribute. The comb routes connecting the net loads to either the fingers or sub-fingers are global routes.

The fishbone trunk and fingers are routed on the two highest layers configured for the net. If sub-fingers are used, they are routed on the layer directly below the finger layer. The comb routes can use any available routing layer for the net.

Fishbone trunks can optionally be biased toward power or ground nets to act as a shield on one side of the trunk, to improve signal integrity on the fishbone trunks. Biasing is enabled separately for horizontal and vertical trunks by setting a spacing rule between the trunk and the power or ground net using the **cts.routing.fishbone_horizontal_bias_spacing** and/or **cts.routing.fishbone_vertical_bias_spacing** app options.

When biasing is enabled, two additional controls are available. The **cts.routing.fishbone_bias_threshold** app option specifies a minimum trunk length before biasing is attempted. The **cts.routing.fishbone_bias_window** app option specifies a maximum displacement from the preferred trunk location to the biased trunk location, above which biasing is not attempted.

Library Cell Subsets

Multisource subtree synthesis honors library cell subsets, if defined. Subsets can be defined in the library manager or the implementation tool by using the **subset_name** attribute, for example:

```
set_attribute [get_lib_cells */INV1*] subset_name INV1
set_attribute [get_lib_cells */INV2*] subset_name INV2
```

Library cells can be resized only within their subset. In the previous example, the */INV1* library cells can be resized to other */INV1* library cells and the */INV2* library cells can be resized to other */INV2* library cells. All inverters that do not belong to one of these subsets can be resized among each other.

Merging is restricted to subsets, so if two cells would otherwise be merge candidates but their library cells belong to different subsets, they are not merged.

Splitting is not affected by subsets, as the split cells all initially get the same library cell as the unsplit reference cell.

Multicorner-Multimode Support

This command works on all the active corners for the given clock mode under an app option cts.multisource.enable_multi_corner_support if set to true. This feature is false by default which means this command works on primary corner of the given clock mode.

EXAMPLES

The following example uses multisource subtree synthesis on all defined clocks with driver objects defined on them.

```
prompt> synthesize_multisource_clock_subtrees
```

The following example calls multisource subtree synthesis on only clocks clk1 and clk2.

```
prompt> synthesize_multisource_clock_subtrees -clocks [get_clocks {clk1 clk2}]
```

The following example calls only the optimize step of subtree synthesis on clock clk.

```
prompt> synthesize_multisource_clock_subtrees -clocks clk -from optimize \
  -to optimize
```

The following example lists the valid steps that make up the multisource subtree synthesis command.

```
prompt> synthesize_multisource_clock_subtrees -list_only
```

SEE ALSO

set_multisource_clock_subtree_options(2)
remove_multisource_clock_subtree_options(2)
report_multisource_clock_subtree_options(2)
derive_clock_balance_points(2)
set_freeze_ports(2)
set_dont_touch(2)
set_size_only(2)
set_placement_status(2)
cts.multisource.ignore_drc_on_subtree_driver(3)
cts.multisource.subtree_merge_cell_name_prefix(3)
cts.multisource.subtree_merge_cell_name_suffix(3)
cts.multisource.subtree_merge_concatenate_length_threshold(3)
cts.multisource.subtree_split_cell_name_prefix(3)
cts.multisource.subtree_split_cell_name_suffix(3)
cts.multisource.subtree_split_net_name_prefix(3)
cts.multisource.subtree_split_net_name_suffix(3)
cts.multisource.subtree_synthesis_enable_power_optimization(3)
cts.multisource.subtree_routing_mode(3)
cts.multisource.verbose(3)

cts.routing.fishbone_max_tie_distance(3)
cts.routing.fishbone_max_sub_tie_distance(3)
cts.routing.fishbone_horizontal_bias_spacing(3)
cts.routing.fishbone_vertical_bias_spacing(3)
cts.routing.fishbone_bias_threshold(3)
cts.routing.fishbone_bias_window(3)

synthesize_multisource_clock_taps

Performs multisource CTS tap assignment.

SYNTAX

status **synthesize_multisource_clock_taps**
[-clocks *clock_list*]

Data Types

clock_list collection

ARGUMENTS

-clocks *clock_list*

Specifies for which clocks tap assignment should be performed. The tap assignment options are defined by the **set_multisource_clock_tap_options** command. By default, all clocks which have tap assignment options defined will be considered.

DESCRIPTION

Regular multisource CTS is an alternative CTS approach which is a hybrid between a traditional automated CTS solution and a more manual full clock mesh implementation (structural multisource). Regular multisource CTS provides benefits such as better OCV tolerance than traditional CTS, and better power than a full mesh flow. Compared to a structural mesh approach, regular multisource trees typically have a less dense mesh structure with fewer tap points from the mesh to the driven taps. Also, the subtrees below the tap drivers can be deeper; structural mesh flows typically have no more than two to three levels beyond the mesh, whereas a regular multisource flow has no restriction on the levels beyond the mesh.

A regular multisource CTS tree has several components, including the pre-mesh tree driving the mesh, the mesh itself, the tap drivers which are driven by the mesh, and the subtrees driven by those tap drivers. The **synthesize_multisource_clock_taps** command is used to assign clock sinks to tap drivers by cloning intermediate clock cells.

Once tap assignment has been done, regular clock tree synthesis and optimization using the **synthesize_clock_trees** or **clock_opt** command is used to synthesize clock subtrees below the tap drivers.

Specifying Tap Assignment Options

Tap assignment options should first be defined by using the **set_multisource_clock_tap_options** command. This command performs two main functions:

- Defines tap driver objects for sink assignment

- Defines a clock to be used for sink identification

A tap assignment driver object set indicates that a set of clock fanouts and sinks are equivalent. Equivalent fanouts and sinks can be freely assigned to any tap driver from the set.

Each tap driver can be synthesized for only one clock. The clock needs to be specified with the **set_multisource_clock_tap_options** command.

Executing Tap Assignment

Tap assignment is performed with the **synthesize_multisource_clock_taps** command. By default, this command operates on all clocks that have tap assignment options defined by the **set_multisource_clock_tap_options** command.

Tap assignment consists of two main steps. First, all clock logic in the fanout of the tap drivers is merged as much as possible and assigned to a single tap driver of that group. Merging all clock logic helps ensure that the subsequent splitting of clock tree logic is done optimally, based on the locations of the tap drivers and the clock sinks.

Merging is done for all equivalent clock cells. The cells need to share common driver nets on each input pin to be considered equivalent for merging. Any timing constraints on the cells need to be the same for merging to occur.

Merging can occur across logical hierarchy boundaries, and merging honors any frozen ports defined by the **set_freeze_ports** command. **dont_touch** constraints on cells and nets in the clock tree can restrict merging. Any cells set as **size_only** or **fixed** are not be merged. Merging can only occur across cells belonging to the same *lib_cell subset* and which are within the same power domain, voltage area, and move bound.

You can prevent merging of cells by specifying them with the **-dont_merge_cells** option of the **set_multisource_clock_tap_options** command. Note, however, that cells specified with this option can still be split (see next step below). This means that a second execution of this command cannot merge these cells again unless the option set definition is altered and the cells are removed from the **-dont_merge_cells** list.

You can allow splitting of cells with user_dont_touch/user_size_only by specifying them with the **-relax_split_restrictions_for_cells** option of the **set_multisource_clock_tap_options** command. These cells will be split for optimization during tap assignment as required, similar to any other cell without user_dont_touch/user_size_only. Note, these cells once split, will not be merged back if **synthesize_multisource_clock_taps** is called consecutively. Hence, user needs to ensure to use a pre-tap assignment block when exploring tap driver locations using multiple calls to **synthesize_multisource_clock_taps**.

The naming convention for the merged cells can be configured through the following app options: cts.multisource.tap_merge_cell_name_prefix, cts.multisource.tap_merge_cell_name_suffix, and cts.multisource.tap_merge_concatenate_length_threshold

After merging is complete, the next step in tap assignment occurs, which is splitting of clock cells to reassign clock sinks to tap drivers. Sinks are assigned and reconnected to the closest tap driver. By default, the distance between a sink and a tap driver is computed by estimating the path length considering the placeable standard cell area. Macros and placement blockages will cause detouring estimations in this mode. By setting the application option **cts.multisource.tap_synthesis_route_based_estimation** to *true* tap assignment will use a buffer-aware global route based distance estimation. This mode uses more runtime but might deliver better assignments in terms of final clock tree synthesis QoR.

The tap drivers need to have a fixed location and placed legally. Placement blockages are honored and macros taken into account while computing the shortest distance to make sure that buffers or other clock logic can be placed along the path from the tap driver to the sink.

Splitting honors frozen ports set with the **set_freeze_ports** command, **set_dont_touch** constraints, **set_size_only** constraints, and fixed placement attributes on cells. Splitting is not performed in case ports would have to be punched on power domain boundaries.

Splitting copies timing constraints, UPF constraints, and user attributes on cells as needed. For example, if a false path is defined through some of the split clock logic, the false path constraint is duplicated to all of the split cells as well. The cloned cells are placed in the same logical hierarchy and voltage area as the original cell.

The naming convention for the cells and nets created during splitting can be configured through the following app options: **cts.multisource.subtree_split_cell_name_prefix**, **cts.multisource.subtree_split_cell_name_suffix**,

cts.multisource.subtree_split_net_name_prefix, and **cts.multisource.subtree_split_net_name_suffix**.

Tap assignment generates logfile output, which gives basic guidance about what the command is doing and any problems encountered during execution. You can generate additional, verbose output by setting the **cts.multisource.verbose** application option to 1. This setting is intended for advanced users doing more detailed investigation of the multisource subtree synthesis results.

Tap Drivers

In a regular multisource CTS flow, the subtrees connect to the mesh through tap drivers. These are specified by the **set_multisource_clock_tap_options -driver_objects** command. The tap driver acts as the driver or root of a particular multisource CTS subtree.

Typically, a tap driver is a clock buffer, but the multisource CTS flow allows a lot of flexibility in the tap driver concept. A tap driver can be an ICG output pin or a clock input port. It is required that tap drivers have a legal and fixed location.

lib_cell Subsets

Multisource tap assignment honors *lib_cell subsets*, if defined. Subsets can be defined in the library manager or an implementation tool by setting the *subset_name* attribute, for example:

```
set_attribute [get_lib_cells */INV1*] subset_name INV1
set_attribute [get_lib_cells */INV2*] subset_name INV2
```

lib_cells can only be resized within their subset. In the above example, **/INV1** lib_cells can be resized to other **/INV1** lib_cells, **/INV2** lib_cells can be resized to other **/INV2** lib_cells, and all inverters not belonging to one of those subsets can be resized among each other.

Merging is restricted to subsets, so if two cells would otherwise be merge candidates but their lib_cells belong to different subsets, they are not merged.

Splitting is not affected by subsets, as the split cells all initially get the same lib_cell as the unsplit master cell.

Sink Groups

Multisource tap assignment allows the user to control individual sink to tap driver assignment using sink groups. Sink groups can be defined using the **create_multisource_clock_sink_group** command. A sink group contains a set of sink pins which should be assigned together to one particular tap driver. A sink group can be either exclusive or non-exclusive. Exclusive means that no other sink which is not in that sink group can be assigned to the tap driver of that sink group. A non-exclusive sink group allows other sinks to get assigned to the tap driver of the sink group as well.

Multivoltage support

By default, tap assignment does not punch ports on power domain boundaries while splitting cells. It also does not split power management cells such as isolation and level shifter cells by default. When the app option **cts.multisource.enable_full_mv_support** is set to *true* then port punching on power domain boundaries and cloning of power management cells is allowed. UPF constraints are honored and copied as required during this. In this case the command also considers the drivability of the tap drivers from an MV point of view. If for example, a tap driver which is geometrically closer to the sinks is less always-on than a given first-level clock cell, then this clock cell cannot be reassigned to that tap driver. In such a case the closest tap driver which can electrically drive the loads is selected. The warning message CTS-741 is printed in case a sink cannot be assigned to its closest tap driver due to this MV check.

Integrated tap assignment in place_opt

When **place_opt.flow.enable_multisource_clock_trees** options is set to *true* during place_opt, **place_opt** will run tap assignment internally and user need not run **synthesize_multisource_clock_taps** explicitly before **clock_opt**. While using this feature, care must be taken that all multisource CTS related setup such as global clock tree (Htree) creation, clock mesh creation, definition of options for tap assignment, tap driver insertion, removing constraints like user_dont_touch, fixed, etc. to allow cell cloning and required CTS related settings are provided before place_opt. With this **place_opt integrated tap_assignment** feature, place_opt can see the clock structure changes done as part of tap assignment, especially ICG clones, and perform improved placement and

optimization. This feature supports all the features currently honored by **synthesize_multisource_clock_taps**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example performs tap assignment on all defined clocks with tap driver objects defined on them.

```
prompt> synthesize_multisource_clock_taps
```

The following example calls multisource tap assignment on only clocks clk1 and clk2.

```
prompt> synthesize_multisource_clock_taps -clocks [get_clocks {clk1 clk2}]
```

SEE ALSO

set_multisource_clock_tap_options(2)
remove_multisource_clock_tap_options(2)
report_multisource_clock_tap_options(2)
create_multisource_clock_sink_group(2)
remove_multisource_clock_sink_groups(2)
report_multisource_clock_sink_groups(2)
get_multisource_clock_sink_groups(2)
add_to_multisource_clock_sink_group(2)
remove_from_multisource_clock_sink_group(2)
synthesize_clock_trees(2)
set_freeze_ports(2)
set_dont_touch(2)
set_size_only(2)
set_placement_status(2)
cts.multisource.subtree_merge_cell_name_prefix(3)
cts.multisource.subtree_merge_cell_name_suffix(3)
cts.multisource.subtree_merge_concatenate_length_threshold(3)
cts.multisource.subtree_split_cell_name_prefix(3)
cts.multisource.subtree_split_cell_name_suffix(3)
cts.multisource.subtree_split_net_name_prefix(3)
cts.multisource.subtree_split_net_name_suffix(3)
cts.multisource.tap_synthesis_route_based_estimation(3)
cts.multisource.enable_full_mv_support(3)
cts.multisource.verbose(3)

synthesize_multisource_global_clock_trees

Performs synthesis and detail routing on specified nets to build H-tree style global clock tree with minimum skew between the endpoints of the net; which is essential in multisource clock tree structure.

SYNTAX

status **synthesize_multisource_global_clock_trees**
[-nets *net_list*]
[-lib_cells *libcell_list*]
[-prefix *name*]
[-roots *pin_port_list*]
[-leaves *pin_list*]
[-skip_pin_connections]
[-use_zroute_for_pin_connections]

Data Types

net_list collection
libcell_list collection
name string
pin_port_list collection
pin_list collection

ARGUMENTS

-nets *net_list*

Specifies the nets for which synthesis and detail routing should be performed. Each specified net is synthesized individually. It is assumed that the leaf pins of each net are regulary distributed such that a symmetric H-tree can be constructed. This option is mutually exclusive to the options *-roots* and *-leaves*. It requires option *-lib_cells*.

-lib_cells *libcell_list*

Specifies a list of library cells of type buffer or inverter which are allowed to be used during synthesis. They need not have any purpose assigned but should not be marked dont_touch. In case multi voltage support is required (which is given when app option *cts.multisource.enable_full_mv_support* is set to *true*) and single and dual rail versions of the repeaters are given the order of the given library cells play a role. The first, second etc. occurrence of a single rail buffer/inverter is matched with first, second etc. occurrence of a dual rail buffer/inverter. It is assumed that these pairs of repeaters are electrical equivalent in terms of input capacitance and delay. Any single or dual rail repeater which is not matched with a counterpart is dropped. This option is mutually exclusive to the options -roots and -leaves. It requires option -nets.

-prefix *name*

Specifies a string which is used as a name prefix for newly created instances during synthesis. This option is mutually exclusive to the options *-roots* and *-leaves*. It requires option *-nets*.

-roots *pin_port_list*

Specifies a list of starting pins or ports in the netlist for which the nets should be detailed routed in H-tree style. It requires option *-leaves*. This option is mutually exclusive to the options *-nets* and *-lib_cells*.

-leaves *pin_list*

Specifies a list of pins in the netlist up to which the nets should be detail routed in H-tree style. It requires option *-roots*. This option is mutually exclusive to the options *-nets* and *-lib_cells*.

-skip_pin_connections

If specified the routing engine will not finish the connection down to the pin shape. It will stop at the highest available metal layer close to the pin shape. By default the routing engine connects all the way down to the pin shapes. However, it is recommended to use the Zroute engine for pin connections.

-use_zroute_for_pin_connections

If specified the Zroute engine is used to finish the connection down to the pin shape. By default, the Custom Router engine is used to do the routing all the way down to the pin shapes. This option will have no effect if *-skip_pin_connections* is specified. It is recommended to use the Zroute engine for pin connections.

DESCRIPTION

The *synthesize_multisource_global_clock_trees* command can be called in two flavors. When called with options *-nets* and *-lib_cells* it will insert repeaters instantiated from library cells specified with option *-lib_cells* on the nets specified with option *-nets* such that the given design constraints are fulfilled. Each net behind the inserted repeaters will be detail routed using the Custom Router. The repeaters are inserted in such a way that the detail routing will have H-tree style. This requires that the sink pins in the nets specified with option *-nets* are distributed regularly in the design area for example by using the **create_clock_drivers** command.

When command *synthesize_multisource_global_clock_trees* is called with options *-roots* and *-leaves* all nets in the netlist between the specified root pins and leaf pins are routed in H-tree style if possible. It is required that each root pin drives the same number of leaf pins and that each such path from root pin to leaf pin has the same number of gate levels. Every net at the same gate level is handled in such a way that the resulting detail routing is as symmetric as possible against the other nets in the same gate level. The Multi-Level Physical Hierarchy Flow (MLPH Flow) is also supported in this mode. Using this flow the H-tree routing can also be done inside the physical hierarchy blocks. MLPH flow can be enabled by setting the application option *cts.multisource.enable_mlph_flow* to true.

In the MLPH flow, the physical blocks must have a fixed placement status, should be editable and port-punching should also be enabled. The optimization of physical blocks must be enabled using the application option *top_level.optimize_subblocks*. In addition, the MSCTS full MV flow should also be turned ON using the application option *cts.multisource.enable_full_mv_support*. To enable the routing to take place inside physical hierarchy blocks, the application option *custom.route.block_soft_macros* must be set to false for the current block.

The router called from command *synthesize_multisource_global_clock_trees* to create the long distance routing is constraint to use the highest two metal layers available for the nets passed with option *-nets* or connecting the cells derived from the cones spanned by the elements of the options *-roots* and *-leaves*. These layers can be controlled by using the commands *set_clock_routing_rules*/*set_routing_rule* in conjecture with an explicit non-default routing rule created with command *create_routing_rule*. The layers should be within global min/max layers specified with *set_ignored_layers*.

The command *synthesize_multisource_global_clock_trees* is making use of application option *cts.multisource.enable_pin_accessibility_for_global_clock_trees* to control the placement of inserted cells such that they are accessible from the selected Htree routing layers. This ensures good global clock tree (Htree) routing topology and hence the QoR. This can be contra productive in case the specified library cells through option *-lib_cells* are multi-row height cells. In such a case

the application option should set to false.

If the command *synthesize_multisource_global_clock_trees* is used on a design with multi voltage (MV) setup it is required to set the application option *cts.multisource.enable_full_mv_support* to *true*. This enables additional code to ensure that a MV clean input is still clean after adding repeaters in the design.

Use the command *remove_multisource_global_clock_trees* to undo the logical and physical changes of command *synthesize_multisource_global_clock_trees*.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example buffers the root net clk and routes all nets driven by the inserted buffers in H-tree style.

```
prompt> synthesize_multisource_global_clock_trees -nets [get_nets clk] \
   -lib_cells [get_lib_cells */CKBUF*]
```

The following example determines all nets between primary input clk and leaf pins clk_leaf_*/A and detail route them in H-tree style. All nets at the same gate level as being equivalent and the resulting detail routing should be as symmetric as possible.

```
prompt> synthesize_multisource_global_clock_trees -roots [get_pins clk] \
   -leaves [get_pins clk_leaf_*/A]
```

The following example inserts a 4 level buffer tree in a multi-physical hierarchy design and creates H-tree style routing using the MLPH flow.

```
prompt> set_editability -blocks  [get_designs -hierarchical] -value  true
prompt> set_app_options -name top_level.optimize_subblocks -value all
prompt> set_app_options -name opt.common.allow_physical_feedthrough -value true
prompt> set_app_options -name cts.multisource.enable_full_mv_support -value true
prompt> set_app_options -name cts.multisource.enable_mlph_flow -value true
prompt> set bufs [get_lib_cells */CKBUF*]
prompt> create_clock_drivers -loads [get_nets clk] \
                        -lib_cells $bufs \
                        -configuration [list [list -level 1 -prefix TAP1 -boxes {1 1}] \
                                             [list -level 2 -prefix TAP2 -boxes {2 1}] \
                                             [list -level 3 -prefix TAP3  -boxes {2 2}] \
                                             [list -level 4 -prefix TAP4 -boxes {4 4}] \
                                       ]

prompt> set_app_options -block [current_block] -name custom.route.block_soft_macros -value false
prompt> synthesize_multisource_global_clock_trees -roots B1/y -leaves [get_pins -hier TAP4*/a] -use_zroute_for_pin_connections
```

SEE ALSO

create_clock_drivers(2)
remove_clock_drivers(2)
remove_multisource_global_clock_trees(2)

create_routing_rule(2)
set_routing_rule(2)
set_clock_routing_rules(2)
set_ignored_layers(2)
cts.multisource.enable_pin_accessibility_for_global_clock_trees(3)

synthesize_regular_multisource_clock_trees

Performs auto tap synthesis and global clock tree synthesis for regular multisource clock trees. It includes auto tap insertion and buffering global clock tree (Htree) as well as detail routing of the global clock tree (Htree). It takes input from set_regular_multisource_clock_tree_options to guide the synthesis specified by the user.

SYNTAX

status **synthesize_regular_multisource_clock_trees**
[-from tap_synthesis | htree_synthesis]
[-to tap_synthesis | htree_synthesis]
[-clocks *clock_list*]
[-list_only]

Data Types

clock_list collection

ARGUMENTS

-from tap_synthesis | htree_synthesis

Specifies that regular multisource clock tree synthesis should start from the specified step. By default, all the steps are run in order: **tap_synthesis**, and **htree_synthesis**. This option can be used to start from a specific step. For example, **synthesize_regular_multisource_clock_trees -from htree_synthesis** skips the tap synthesis step.

The *-from* and *-to* options can be used together or separately to run only certain steps of the regular multisource synthesis.

-to tap_synthesis | htree_synthesis

Specifies that regular multisource clock tree synthesis should end at the specified step. By default, all the steps are run in order: **tap_synthesis**, and **htree_synthesis**. This option can be used to end at a specific step. For example, **synthesize_regular_multisource_clock_trees -to tap_synthesis** skips the Htree synthesis step.

-clocks *clock_list*

Specifies for which clocks auto tap synthesis should be performed. By default, clocks or nets defined through set_regular_multisource_clock_tree_options will be considered.

-list_only

Lists the available steps for the regular multisource clock tree synthesis command. These steps are **tap_synthesis**, and **htree_synthesis**. By default, all steps are run when the **synthesize_regular_multisource_clock_trees** command is used. But, the *-from* and *-to* options can be used to limit the command to execute only the specified steps. *-list_only* simply lists the valid steps that can be passed to the *-from* and *-to* options.

DESCRIPTION

The *synthesize_regular_multisource_clock_trees* command performs regular multisource clock tree synthesis, from auto tap synthesis to Htree synthesis. It takes input from option command set_regular_multisource_clock_tree_options. The feature aims at automatically creating Htree friendly tap drivers using configuration (tap_boxes) from user as guidance. This helps user to arrive at suitable tap driver configuartion easily.

In auto tap synthesis, it inserts tap drivers based on *-tap_boxes* and library cells from the option *-tap_lib_cells* in the options command and select the locations based on clock sinks distribution. When *-tap_template_cells* is given in the options command, it will split the template cell as tap drivers instead of inserting repeaters. The tap locations are optimized to form a regular grid and also a Htree build feasibility is performed at this step to help find a set of taps that will likely be accepted at global clock tree synthesis step. You can use additional switches -max_displacement, -keepout and -tap_boundary to control max_displacement of tap drivers from referred locations and region allowed for tap driver insertion.

After auto tap synthesis, the command will do the global clock tree synthesis. It will insert repeaters instantiated from library cells specified through the option command on option *-htree_lib_cells* on the nets specified with option *-nets* such that the given design constraints are fulfilled similar to synthesize_multisource_global_clock_trees. Each net in the global clock tree will be detail routed using Custom Router for balanced routing and use zroute for final pin connections. The repeaters are inserted in such a way that the detail routing will have Htree style.

The tap drivers and Htree repeaters by default have "clk_drv" and "msgts" as prefix. You can specify desired naming convention using -prefix switch.

The command synthesize_regular_multisource_clock_trees picks up the Htree NDR and layer list specified through the option command on option -htree_routing_rule and -htree_layer_list. If these are not specified then same are picked up from the clock tree settings specified with set_routing_rule and set_clock_routing_rules. The layers should be within global min/max layers specified with set_ignored_layers.

The command synthesize_regular_multisource_clock_trees makes use of applictaion option cts.multisource.enable_pin_accessibility_for_global_clock_trees to control the placement of inserted cells such that they are accessibe from the selected Htree routing layers. This ensures good global clock tree (Htree) routing topology and hence the QoR.

If the command synthesize_regular_multisource_clock_trees is used on a design with multi voltage (MV) setup it is required to set the application option cts.multisource.enable_full_mv_support to true. This enables additional code to ensure that a MV clean input is still clean after adding repeaters in the design.

Commands report_regular_multisource_clock_tree_options and remove_regular_multisource_clock_tree_options can be used to report and remove the options as required.

Use commands remove_multisource_global_clock_trees and remove_clock_drivers to remove htree and tap drivers as required. These are useful when exploring different tap driver configurations.

For detailed messaging you can cts.multisource.verbose application option during debug.

Once tap drivers are inserted and Htree is built you can define tap assignment settings using set_multisource_clock_tap_options and run tap assignment using synthesize_multisource_clock_taps. If using tap assignment at place_opt then only set_multisource_clock_tap_options is sufficent. Please refer to place_opt.flow.enable_multisource_clock_trees for more details.

To do multiple clock synthesis using this command, the order of the synthesis will follow the order of the clocks under -clocks. If -clocks is not given, the synthesis will follow the order of the set_regular_multisource_clock_tree_options.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example inserts tap drivers in 4 columns and 2 rows for clock **clk** ensuring Htree compatibility and builds Htree automatically to drive the tap drivers using specified library cells, NDR and layer list.

```
prompt> set_regular_multisource_clock_tree_options -clock clk -topology htree_only -prefix MSCTS \
 -tap_boxes {4 2} -tap_lib_cells [get_lib_cells LIB1/BUF1] \
 -htree_lib_cells [get_lib_cells LIB1/BUF2] -htree_layers "m9 m10" -htree_routing_rule "htree_ndr"

prompt> synthesize_regular_multisource_clock_trees
```

Similar to the above example, the following example performs auto tap synthesis and htree synthesis with successive command calls for debug.

```
prompt> set_regular_multisource_clock_tree_options -clock clk -topology htree_only -prefix MSCTS \
 -tap_boxes {4 2} -tap_lib_cells [get_lib_cells */CKBUF*] \
 -htree_lib_cells [get_lib_cells */CKINV*] -htree_layers "m9 m10" -htree_routing_rule "htree_ndr"

prompt> synthesize_regular_multisource_clock_trees -from tap_synthesis  -to tap_synthesis
prompt> synthesize_regular_multisource_clock_trees -from htree_synthesis  -to htree_synthesis
```

SEE ALSO

set_regular_multisouce_clock_tree_options(2)
report_regular_multisouce_clock_tree_options(2)
remove_regular_multisouce_clock_tree_options(2)
remove_clock_drivers(2)
remove_multisource_global_clock_trees(2)
create_routing_rule(2)
set_routing_rule(2)
set_clock_routing_rules(2)
create_clock_drivers
synthesize_multisource_global_clock_trees
set_multisource_clock_tap_options
synthesize_multisource_clock_taps
cts.multisource.enable_full_mv_support
cts.multisource.verbose
cts.multisource.enable_pin_accessibility_for_global_clock_trees
place_opt.flow.enable_multisource_clock_trees

train_pg_ml_model

Create ML training files based on created ML training data in memory, and then train the ML model for fixability prediction.

SYNTAX

status **train_pg_ml_model**
[-input_directory *directory_name*]

Data Types

directory_name string

ARGUMENTS

-input_directory *directory_name*

Specify directory name of created ML training file. If the option is not specified, the command creates ML training files in the default directory ./pg_model based on ML training data in memory.

DESCRIPTION

This command creates ML training files based on created ML training data in memory, and then train the ML model for fixability prediction. The trained model is stored in the directory ./pg_model

EXAMPLES

The following example creates ML training data in memory, output training data to files in directory ./pg_model, and train ML model based on the training data. The trained ML model is stored in the directory ./pg_model.

prompt> **compile_pg -create_ml_data**
prompt> **train_pg_ml_model**

The following example creates ML training data in memory, output training data to files in directory pg_data_1, and train ML model on the training files in the directory pg_data_1. The trained ML model is stored in the directory ./pg_model.

prompt> **compile_pg -create_ml_data**
prompt> **create_pg_ml_data -output_directory pg_data_1**

```
prompt> train_pg_ml_model -input_directory {pg_data_1}
```

SEE ALSO

compile_pg(2)
create_pg_ml_data(2)
create_pg_vias(2)

transform_polygons

Performs translation and rotation of a geometric region and returns the result as a geo_mask.

SYNTAX

collection **transform_polygons**
[-objects *object_list*]
[*pos_object_list*]
[-orientation *orientation*]
[-coordinate {*x y*}]
[-inverse]

Data Types

object_list collection
pos_object_list collection
orientation string
x float
y float

ARGUMENTS

-objects *object_list*

Specifies the objects to be used to define the geometric region to be transfomed. Objects can be a heterogeneous collection of poly_rects, geo_masks, shapes, layers, and other physical objects.

In the case of poly_rects, geo_masks, shapes, or other physical objects, the resulting area will include the areas of each object. In the case of layers, the resulting area will include the area of every shape in the layer.

This is a required option.

-orientation *orientation*

Specifies the orientation to which the geometric region should be rotated. Rotation is counterclockwise about the origin ({0 0}). Valid value for the orientation are: R0, R90, R180, R270, MX, MXR90, MY, MYR90.

-coordinate *coordinate*

Specifies the relative coordinate to translate (shift) the geometric region by. If the *-orientation* option is also specified, the translation will happen after the rotation.

-inverse

This argument when provided with transform_polygons command, inverts the desired transformation.

DESCRIPTION

This command performs translation and/or rotation of a geometric region specified by *objects* and returns the result as a geo_mask object. The objects in *objects* are not modified, rather the result of the transformation of the region specified by *objects* is returned.

EXAMPLES

The following example returns a geo_mask with a rectangle rotated 90 degrees counterclockwise and moved up by 10 units.

```
prompt> transform_polygons -coordinate {0 10} -orientation R90 \
  -objects [create_poly_rect -boundary {{100 100} {130 110}}]
```

SEE ALSO

compute_area(2)
copy_to_layer(2)
create_geo_mask(2)
create_poly_rect(2)
split_polygons(2)
resize_polygons(2)

trim_pg_mesh

Trims dangling wires of PG nets after the PG mesh is created.

SYNTAX

status **trim_pg_mesh**
[-nets *netname_list*]
[-types *type_list*]
[-layers *layer_list*]
[-trim_to *trim_to_opt*]
[-drc *drc_opt*]
[-verbose *verbose_opt*]
[-shapes *shape_list*]
[-undo]

Data Types

netname_list list
type_list list
layer_list list
trim_to_opt specification
drc_opt specification
verbose_opt specification
shape_list list

ARGUMENTS

-nets *net_list*

Specifies a list of PG nets on which to trim wires. By default, all PG nets are processed.

-types *type_list*

Specifies a list of PG object types. The PG types should include one or more of the following keywords: **ring**, **stripe**, **lib_cell_pin_connect**, **macro_pin_connect**, and **macro_conn**. These types are for PG rings, PG straps, standard cell pin connections, macro pin connections. By default, all of these types are considered for wire trimming.

-layers *layer_list*

Specifies metal layer names for the PG objects. *layer_list* is a list containing layer names. By default, objects on all metal layers are considered for wire trimming.

-trim_to target_wire | via

Specifies the stop criteria for wire trimming. If **via** is specified, then wire ends are trimmed to the closest same net and same layer

via metal enclosure. By default, wire ends are trimmed to closest target of another same net PG wires, pins, or terminals.

-drc no_check | **check_but_no_fix**

Specifies the DRC option for PG wire trimming. The argument following **-drc** must be either no_check or check_but_no_fix. no_check means that no DRC check is performed. check_but_no_fix means that DRC check is performed, DRC errors are reported, but no DRC fixing is performed. By default, DRC is checked and fixed for the PG wire trimming. If wire has DRC and the tool cannot fix it, the wire is restored to an original shape.

-verbose none | **{dangling drc}**

Writes out extra information to the log file. The argument following **-verbose** must be either **none** or one of: **dangling**, **drc** or **{dangling drc}**. If **dangling** is specified, **gui_add_annotation** commands are written to the log file with the original shape bounding box of the deleted dangling wire. If **drc** is specified, **gui_add_annotation** commands are output into log file with original shape bounding box of restored due to DRC wire in addition to DRC violation report. Default is **none**

-shapes ***shape_list***

Specifies a list of PG shapes for which to trim wires. By default, all feasible wires of PG nets will be trimmed.

-undo

Restores trimmed shapes to their original dimensions, and restores deleted dangling wires and vias, trimmed by most recent **trim_pg_mesh** command. Only one level of undo is supported; the **trim_pg_mesh -undo** command clears the undo command stack.

DESCRIPTION

This command performs PG wire trimming after the PG mesh is created for the whole design. This command is used to remove dangling wire ends and trim them to the closest target of other same net PG wires or pins or terminals. If **trim_to via** is specified, the wire ends are trimmed to the closest same net and same layer via metal enclosure. If the tool finds dangling wires, the wire and connecting vias are removed. The criteria for dangling wire is a wire, connecting to just one same or different layer target with via or via stack with one or both wire ends dangling (not connected). The tool may leave some dangling wires to avoid creating a DRC violation if the wire may have been routed as a patch to cover a minimum edge length, minimum width, minimum area, or other DRC violation.

The **trim_pg_mesh** command can be run several times to further trim wires after dangling shapes are removed. Note that you can only restore shapes to the state before most recent **trim_pg_mesh** command; only one level of undo is supported. Floating wire shapes and vias, which represent a disconnected component of any number of wires and vias, should be removed using the **check_pg_connectivity** PG command. This command returns a collection of floating vias and wires. You can easily remove floating objects reported by **check_pg_connectivity** with the **remove_objects command**. You should remove any floating PG objects before running **trim_pg_mesh**.

EXAMPLES

The following example performs PG wire trimming on PG nets VDD and VSS after PG mesh is created with one of PG creation commands.

```
prompt> compile_pg
prompt> trim_pg_mesh \
   -nets {VDD VSS}
```

The following example performs PG wire trimming on all PG nets, only for stripes and standard cell connections on layer M5. DRC is checked but not fixed, and DRC report and bounding boxes of wires with DRC and dangling wires are output into log.

```
prompt> compile_pg
prompt> trim_pg_mesh \
   -types {stripe lib_cell_pin_connect} \
   -layers M5 \
   -drc check_but_no_fix \
   -verbose {dangling drc}
```

SEE ALSO

check_pg_connectivity(2)
compile_pg(2)
create_pg_composite_pattern(2)
create_pg_macro_conn_pattern(2)
create_pg_ring_pattern(2)
create_pg_std_cell_conn_pattern(2)
create_pg_strap(2)
get_shapes(2)
get_vias(2)
remove_pg_patterns(2)
report_pg_patterns(2)
report_pg_strategies(2)
set_host_options(2)
set_pg_strategy(2)

trim_shapes

This command symmetrically trims line-ends of shapes according to specified line-end spacing values per layer.

SYNTAX

status **trim_shapes**
-line_end_spacings {*list_of_layer_value_pairs*}

Data Types

list_of_layer_value_pairs list

ARGUMENTS

-line_end_spacings {*list_of_layer_value_pairs*}

Specifies the desired line-end spacing per layer.

{*layerName value*}

DESCRIPTION

This command symmetrically trims line-ends of shapes according to specified line-end spacing values per layer. For line-end spacings greater or equal to the specified layer value, nothing is done. For line-end spacings smaller than the specified value, both line-ends are trimmed symmetrically such that the specified spacing is accomplished between the new line-ends. For line-ends which are defined by a via surround, the via might be replaced by one with a smaller surround to accomplish the desired spacing. Please note that vias which are excluded from signal routing (excludedForSignalRoute = 1) are also used for replacement.

Only wires and vias can be trimmed (signal, PG and clock nets), other situations remain unchanged. All situations which could not be fixed are flagged as DRC violations. Other DRC types are not checked or reported.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

prompt> **trim_shapes -line_end_spacings {{M1 0.02} {M2 0.03}}**

unalias

Removes one or more aliases.

SYNTAX

string **unalias**
patterns

ARGUMENTS

patterns

Specifies the patterns to be matched. This argument can contain more than one pattern. Each pattern can be the name of a specific alias to be removed or a pattern containing the wildcard characters ***** and **%**, which match one or more aliases to be removed.

DESCRIPTION

The **unalias** command removes aliases created by the **alias** command.

EXAMPLES

The following command removes all aliases.

prompt> **unalias ***

The following command removes all aliases beginning with f, and the alias rt100.

prompt> **unalias f* rt100**

SEE ALSO

alias(2)

uncommit_block

Merges the contents of a physical block into the parent design and updates the reference of the instance, and removes the design for the block if specified.

SYNTAX

collection **uncommit_block**
-design *reference_design*
-type *cell_hierarchy_type*
[-remove_design]
[-verbose]

Data Types

reference_design string or collection
cell_hierarchy_type string

ARGUMENTS

-design *reference_design*

Specifies the name of the cell reference design to be uncommitted. This is a required option.

-type *cell_hierarchy_type*

Specifies the cell hierarchy type after uncommit. Valid types are **module_boundary** and **module**. By default, the type is **module_boundary**.

-remove_design

Removes the design referenced by cell.

-verbose

Displays additional debugging information.

DESCRIPTION

This command removes the physical hierarchy for the specified design and converts the reference into a logical hierarchy cell. The command fails if the given cell path name or object does not exist or is already a logical hierarchy cell.

If the **-remove_design** option is used, the command removes the reference design for the specified cell.

As part of the uncommit process, certain physical elements within the block are propagated to the parent cell. Among these elements are routing, route guides, placement blockages, move bounds and relative placement groups.

Also, logical group bounds are propagated automatically to the top block during uncommit_block.

If there is name conflict for relative placement group then its name is prefixed with name of the uncommitted block.

If the block to be uncommitted has site arrays, they are propagated up to the parent if they are non-default site arrays and they have no conflicts with existing non-default site arrays in the parent.

Default site arrays are not propagated up. But the command does check block default site arrays and confirms that if the block has a default site array, the command asserts that the parent must have a default site array also. If it doesn't the command will issue an error and reject the block for uncommit. If a default site array exists in both the block and the parent, the command asserts that they must be identical in definition and that the default rows in the block are aligned with the ones in the parent. If there is any type of incompatibility between the default site array in the block and in the parent, the command rejects the block for uncommit. There is, however, an exception to this rule. If a block being uncommitted is specified to be uncommitted to a module (no hier boundary) AND the reference block has no cells that are placed, then the checking rule is relaxed for default site arrays consistency with the top.

The changes to the netlist, as a result of uncommitting blocks to modules, are dramatic, and structural. Because of this, it is necessary to purge ALL existing timing constraints before uncommitting the blocks. This is done automatically inside of uncommit_block. This will affect all top-level timing constraints, including attributes such as "dont_touch" and other timing-based attributes. It is necessary for the user to re-apply timing constraints after running this command.

Please note, also, that if uncommit_block is performed in a "bottom-up" approach (where a mid-level block of a MPH design is opened as the current block), that the automatic purging of timing constraints in the current block will result in linking problems when the top block is subsequently opened.

EXAMPLES

This example uncommits the block instances of design A:

```
prompt> uncommit_block -design A
```

SEE ALSO

commit_block(2)
get_cells(2)
set_cell_hierarchy_type(2)

undefine_user_attribute

Removes the specified user-defined attribute from all open libraries.

SYNTAX

status **undefine_user_attribute**
-classes *class_list*
[-quiet]
[-force]
-name *attribute_name*
pos_attribute_name

Data Types

class_list list
attribute_name string
pos_attribute_name string

ARGUMENTS

-classes *class_list*

Specifies the classes from which to remove the user-defined attribute. Valid classes are design, port, cell, net, clock, and so on.

-quiet

Turns off the warning message that would otherwise be issued if the attribute or classes are incorrectly specified.

-force

Forces removal of the user-defined attribute, even if the attribute is set for some objects.

-name *attribute_name*

Specifies the name of the attribute. This option is mutually exclusive with *pos_attribute_name*. Specify either **-name** or *pos_attribute_name*, but not both.

pos_attribute_name

Specifies the name of the attribute. This positional option is provided for compatibility with other tools. This option is mutually exclusive with **-name**. Specify either **-name** or *pos_attribute_name*, but not both.

DESCRIPTION

This command removes the user-defined attribute from all the libraries opened in the session. The attribute is removed from the classes that support user-defined attributes.

The command issues an error message and exits under the following conditions:

- The specified attribute is an application attribute (not a user-defined attribute) in any of the class specified by the **-classes** option.
- The class list contains a class where the specified attribute is set for at least one object in any design.

Use the **list_attributes** command to confirm that the attributes you undefined are removed from the design.

EXAMPLES

The following example sets a user-defined attribute named *attr_1* and undefines it using **-name** option. The attribute is removed from the track class.

```
prompt> define_user_attribute -classes {track} -type double \
  -range_min 2.0 -range_max 3.2 -name attr_1

prompt> undefine_user_attribute -name attr_1 -classes track
1
```

The following example undefines a user-defined attribute named *attr_1* by specifying it as a positional attribute.

```
prompt> undefine_user_attribute attr_1 -classes track
1
```

The following example tries to undefine an application attribute name named *layer* from the track class by using the **-name** option. The command issues an error message and exits because layer is an application attribute, not a user-defined attribute.

```
prompt> undefine_user_attribute -name layer -classes track
Error: Attribute 'layer' is not user-defined for class(es) 'track'.
Cannot undefine it. (ATR-016)
```

The following example tries to undefine a user-defined attribute named *attr_1* by using the **-name** option. The command issues an error message and exits because the attribute is set for at least one object in a design for the track class.

```
prompt> undefine_user_attribute -name attr_1 -classes track
Error: Attribute 'attr-1' has been set for at least one of the object of
following class(es) 'track'. Cannot undefine it. (ATR-018)
```

The following example undefines a user-defined attribute named *attr_1* by using the **-name** option. The first command fails because the attribute is set for at least one object in a design for the track class. The second command uses the **-force** option to force removal of the attribute.

```
prompt> undefine_user_attribute -name attr_t -classes track
Error: Attribute 'attr_t' has been set for object(s) of following
class(es): 'track'. Cannot undefine it. (ATR-018)
```

```
prompt> undefine_user_attribute -name attr_1 -classes track -force
1
```

SEE ALSO

define_user_attribute(2)
get_attribute(2)
get_defined_attributes(2)
list_attributes(2)
remove_attributes(2)
report_attributes(2)
set_attribute(2)

undo

Undo the effects of one or more commands.

SYNTAX

int **undo**
[-levels *num_levels* | -marker *marker_name*]
[-check_only]
[-silent]

Data Types

num_levels int
marker_name string

ARGUMENTS

-levels *num_levels*

Undoes the effects of the specified number of command levels.

-marker *marker_name*

Reverts the system state to the point of creation of the marker with the specified user or system name.

-check_only

Causes the command to return the same status and messages it would otherwise return without actually performing the undo or changing anything.

-silent

Surpresses messages and a TCL error in the event of an error.

DESCRIPTION

This command undoes the effects of one or more previous database-changing commands, or "command levels". If the *-levels* option is used, then the specified number of command levels will be undone. It is an error to specify more levels than there are in the command undo history.

Note that a command level refers to a command that explicitly changes the system state, so the undoing of one level may revert the state back several commands, such that only the earliest command was a database-changing command, the other more recent commands being ignored. Some commands that change environmental parameters but not the database are also exempt and are not considered as command levels. Use the *get_undo_info* command to determine what the command corresponding to the next undoable command level is.

If the *-marker* option is used, then the system state will be reverted to that of when the specified marker was created. *marker_name* may be the user assigned or system generated name of the marker. This option would typically be used to revert back to a user marker, which would have been created using the *create_undo_marker* command. It is an error to specify a marker that was created later than the current point in the command history - for that case use the *redo* command instead.

If neither the *-levels* or *-marker* option is used, then this command will undo one command level.

The command returns the actual number of command levels undone or, if the *-check_only* option was used, that would have been undone without the *-check_only* option.

Not all commands are undoable. If a non-undoable command is executed, the undo history will be cleared. Use the *get_undo_info* command to determine which commands are undoable.

To preserve system resources and minimize the impact on performance, the undo system limits the size of the undo history it maintains. These limits can be adjusted via the following app options: *undo.max_levels* - the maximum number of command levels to save, and *undo.max_memory* - the maximum amount of memory (in bytes) the undo system may use.

EXAMPLES

The following undoes one command level:

```
prompt> get_attribute $inst origin
0.0000 0.0000
prompt> set_attribute $inst origin {100 300}
{U368}
prompt> get_attribute $inst origin
100.0000 300.0000
prompt> undo
1
prompt> get_attribute $inst origin
0.0000 0.0000
```

The following undoes back to a user marker:

```
prompt> create_undo_marker my_mrk
my_mrk
prompt> get_attribute $inst origin
0.0000 0.0000
prompt> set_attribute $inst origin {200 400}
{U368}
prompt> set_attribute $inst origin {300 600}
{U368}
prompt> get_attribute $inst origin
300.0000 600.0000
prompt> undo -marker my_mrk
2
prompt> get_attribute $inst origin
0.0000 0.0000
```

SEE ALSO

redo(2)
create_undo_marker(2)
eval_with_undo(2)
get_undo_info(2)

ungroup_cells

Removes one level of hierarchy.

SYNTAX

status **ungroup_cells**
cell_list | -all
[-flatten]
[-simple_names]
[-prefix *prefix_name*]
[-start_level *n*]
[-force]

Data Types

cell_list list
prefix_name string
n integer

ARGUMENTS

cell_list

Specifies a list of hierarchical cells whose hierarchies will be ungrouped. If more than one cell is specified, the list must be enclosed in braces ({}). You must specify either *cell_list* or the -all keyword, but not both.

-all

Indicates that all hierarchical cells in the current block (or current instance if set) are to be ungrouped. You must specify the *-all* keyword or *cell_list*, but not both.

-flatten

Indicates that the specified cells are to be ungrouped recursively until all levels of hierarchy are removed.

-simple_names

Indicates that simple and nonhierarchical names are to be used for cells that are ungrouped. When this option is not specified, cells are given the default hierarchical names. With this option, cells maintain their original names.

-prefix prefix_name

Specifies the prefix to use in naming ungrouped cells. The default naming style is as follows:

cell_being_ungrouped/*old_cell_name*{*number*}

-start_level n

Causes all hierarchical cells that are at the level specified by n to be flattened. This option implies the **-flatten** option. The level value can be 1, 2, 3, and so on. Specifying a value of 1 flattens the cells from the current design. Specifying a value of 2 maintains the top-level hierarchy and all cells in each of the top-level hierarchy to be flattened.

-force

Causes all dont_touch hierarchical cells to be flattened. The dont_touch attribute is inherited by their leaf cells. This option can only be used when either the **-all** keyword or *cell_list* is specified.

DESCRIPTION

This command removes a single level of hierarchy by dissolving the specified hierarchical cell and absorbing its internal cells and nets into the parent cell. Nets that were connected to ports in the cell that was ungrouped are deleted, and the corresponding nets in the parent cells are extended to connect to the newly promoted cells' pins.

Nets and cells from the dissolved hierarchies are renamed to preserve their original path names from the top block. The new names are formed by joining the instance name of the dissolved cell with the original net or cell name separated by the hierarchical separator character (default '/'). For example, if ungrouping cell mid1 causes cell U8 to be moved to mid1's parent cell, the command renames cell U8 to be mid1/U8. This renaming allows power and timing constraint files to work correctly with the ungrouped objects.

The **ungroup_cells** command does not collapse block boundaries, so the specified cells might not directly instantiate a block. If the *-flatten* option is specified, the submodule recursion will stop at any block boundaries.

Cells marked with the **dont_touch** attribute cannot be ungrouped.

No constraints are moved or deleted by this command. If a specified cell has timing or power constraints on itself or its pins, it cannot be ungrouped.

This command operates on a single instance of the parent module that contains the cell being ungrouped. If that module is multiply instantiated, a separate and unique module will be created for this instance and the cell will be ungrouped only in this instance module.

EXAMPLES

The following example dissolves the hierarchical cells mid1/bot1 and mid1/bot2 and group their cells into module mid1:

```
prompt> get_cells mid1/*
{mid1/bot1 mid1/bot2}
prompt> ungroup_cells [get_cells mid1/*]
1
prompt> get_cells mid1/*
{mid1/bot2/c1 mid1/bot1/c1 mid1/bot1/c2 mid1/bot2/c2}
```

The following example dissolves all hierarchy below the current instance:

```
prompt> ungroup_cells -flatten -all
1
```

SEE ALSO

group_cells(2)
set_dont_touch(2)

uniquify

Creates individual reference copies for cells that have logic module references in the current design.

SYNTAX

collection **uniquify**
[cell_list]
[-verbose]
[-uniquify_children]
[-force]
[-base_name *base_name*]
[-new_name *new_name*]

Data Types

cell_list list
base_name string
new_name string

ARGUMENTS

cell_list

Specifies a list of hierarchical cells to be bound to unique reference copies. Each cell must be in the scope of the current instance, that is, in the hierarchy. Each cell must have a logic module reference.

-verbose

Prints additional messages.

-uniquify_children

Includes child hierarchical instances for the uniquify process. Instances with references of type block or lib_cell are stopping points.

-force

Uniquifies even if they are already unique or are marked with the dont_touch attribute. The top-level design and the ILMs are not renamed.

-base_name *base_name*

Specifies the base name to be used instead of the original module name for the new module.

-new_name *new_name*

Specifies the module name to use for a single cell specified using cell_list. If **new_name** conflicts with an existing module name, **design.uniquify_naming_style** is used to create new module name. This option can not be used with -force or -uniquify_children.

DESCRIPTION

The **uniquify** command takes the reference module of each cell in the current design and creates a unique copy. Then, all cells point to their individual reference modules. If the dont_touch attribute is set on a cell or a module, a new module is not created for that cell or any cell with that reference module. If a list of hierarchical cells is specified, the **uniquify** command works on the complete paths to these specified cells. A unique reference module is generated for each cell in the specified list even if the dont_touch attribute is set on the cell or its reference module. All the parent cells of the specified cells are also uniquified if they are not uniquified already.

A cell list might include cells that have different references. When a cell list is provided by the **get_cells** command, it automatically uses instances that have logic module references and prints out warnings for instances that do not have logic module references. The name of the new reference copy is determined by the value of the **design.uniquify_naming_style** application option that is set for the parent design. The default is "%s_%d".

Note: Empty modules are uniquified by default; these can be skipped by setting the **design.uniquify_skip_empty_modules** application option to true.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates uniquely named reference modules for all cells in the current design, but not for cells or their reference modules marked with the dont_touch attribute:

```
prompt> uniquify
```

In the following example, the **uniquify** command creates unique reference copies for the specified cells:

```
prompt> uniquify {u2 u3}
2
```

In the following example, u1 and u2 are instances of unique references mid1 and mid2. Executing the uniquify -force or uniquify {u1 u2} command renames the reference copies:

```
prompt> uniquify {u1 u2}
2
prompt> get_modules
{mid1_0 mid2_0}
```

In the following example, a new module "ADDER1" is created for cell u4 in the current design.

```
prompt> uniquify {u4} -new_name ADDER1
1
```

SEE ALSO

commit_block(2)
change_reference(2)
set_reference(2)
uncommit_block(2)
uniquify_block(2)
design.uniquify_naming_style(3)
design.uniquify_skip_empty_modules(3)

uniquify_block

Create individual reference copies for specified cells having block references.

SYNTAX

collection **uniquify_block**
[-library library]
cell_list

Data Types

cell_list list
library library name

ARGUMENTS

cell_list

Specifies the list of hierarchical cells to be bound to unique reference copies. Each cell must be in scope (at or below the current instance). Each cell must have block reference.

-library

Specifies the library name in which to create the copy of the block reference. Library provided should be an existing one. If this option is not specified, new copy will be created in library of existing reference.

DESCRIPTION

The **uniquify_block** command creates individual copies of block references for each specified cell. While making copy of block reference, all views are copied. Cells are then bounded to their respective views. The existing reference could only be a block in the current design. Cell list may include cells having difference references. When cell list is provided through **get_cells**, command automatically picks up instances that have block reference and print out warning for instances not having block reference. Name of the new reference copy will be determined using value of app option "design.uniquify_naming_style" set for the parent design. Default value is "%s_%d". This command shall implicitly retain the editability of the old reference by propagating the hierarchical edit-control to the new reference.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, uniquify_block is used to create unique reference copies for cells provided. New reference copies of blocks are created in library lib1.

```
prompt> uniquify_block -library lib1 {u2 u3}
2
```

SEE ALSO

commit_block(2)
change_reference(2)
set_reference(2)
uncommit_block(2)
uniquify(2)
set_editability(2)
get_editability(2)
report_editability(2)
design.uniquify_naming_style(3)

unplace_group_repeaters

Unplace the previous placement of the repeaters.

SYNTAX

status **unplace_group_repeaters**
-cells *cell_list* |
-repeater_groups *group_id_list*
[-lib_cell_input *pin_name*]
[-lib_cell_output *pin_name*]

Data Types

cell_list	collection
group_id_list	list
pin_name	string

ARGUMENTS

-cells *cell_list*

Specifies the list of repeaters to be unplaced in the pipeline repeater paths. The two options -cells and -repeater_groups are mutually exclusive, and one of them is required.

-repeater_groups *group_id_list*

Specifies the list of repeater group ids. The two options -cells and -repeater_groups are mutually exclusive, and one of them is required.

-lib_cell_input *pin_name*

Specifies the pin name when lib cell has multiple input pins. The default value is D.

-lib_cell_output *pin_name*

Specifies the pin name when lib cell has multiple output pins. The default value is Q.

DESCRIPTION

This command unplaces the previous placement of the repeaters on route. The routes of the connected nets of the unplaced registers will be merged to complete route recovery.

This command unplaces only those repeaters that were added using add_group_repeaters or placed using place_group_repeaters.

However, for the virtual connections (that is, for the cells between groups that may not be connected directly), you need to set the app option eco.placement.eco_enable_virtual_connection to true.

EXAMPLES

The following examples show the usages of command.

The following examples show the usages of -cells.

```
prompt> unplace_group_repeaters -cells $ecoCells -lib_cell_input D -lib_cell_output Q
```

The following examples show the usages of -repeater_groups.

```
prompt> set_repeater_group -group_id 1 -cells [get_cells {eco_cell_3 eco_cell_7}] -cutline {{720 366} {720 386}}
prompt> set_repeater_group -group_id 2 -cells [get_cells {eco_cell_2 eco_cell_6}] -cutline {{920 366} {920 386}}
prompt> set_repeater_group -group_id 3 -cells [get_cells {eco_cell_1 eco_cell_5}] -cutline {{1230 330} {1230 350}}
prompt> set_repeater_group -group_id 4 -cells [get_cells {eco_cell_0 eco_cell_4}] -cutline {{1430 330} {1430 350}}
prompt> unplace_group_repeaters -repeater_groups {1 2 3 4} -lib_cell_input D -lib_cell_output Q
```

SEE ALSO

add_group_repeaters(2)
place_group_repeaters(2)

unsetenv

Removes a system environment variable.

SYNTAX

string **getenv**
variable_name

Data Types

variable_name string

ARGUMENTS

variable_name

Specifies the name of the environment variable to be unset.

DESCRIPTION

The **unsetenv** command searches the system environment for the specified *variable_name* and removes variable from the environment. If the variable is not defined in the environment, the command returns a Tcl error. The command is catchable.

Environment variables are stored in the **env** Tcl array variable. The **unsetenv**, commands is a convenience function to interact with this array. It is equivalent to 'unset ::env(*variable_name*)'

The application you are running inherited the initial values for environment variables from its parent process (that is, the shell from which you invoked the application). If you unset the variable using the **unsetenv** command, you remove the variable value in the application and in any new child processes you initiate from the application using the **exec** command. However, the variable is still set in the parent process.

See the **set** and **unset** commands for information about working with non-environment variables.

EXAMPLES

In the following example, **unsetenv** remove the DISPLAY varible from the environment:

prompt> **getenv DISPLAY**

```
host:0
prompt> unsetenv DISPLAY
prompt> getenv DISPLAY
Error: can't read "::env(DISPLAY)": no such variable
    Use error_info for more info. (CMD-013)
```

SEE ALSO

catch(2)
exec(2)
printenv(2)
set(2)
unset(2)
setenv(2)
getenv(2)

unsuppress_message

Enables printing of one or more suppressed informational or suppressed warning messages.

SYNTAX

string **unsuppress_message** [*messages*]

list *messages*

ARGUMENTS

messages

A list of messages to enable.

DESCRIPTION

The **unsuppress_message** command provides a mechanism to re-enable the printing of messages which have been suppressed using **suppress_message**. You can suppress only informational and warning messages, so the **unsuppress_message** command is only useful for informational and warning messages. The result of **unsuppress_message** is always the empty string.

You can suppress a given message more than once. So, you must unsuppress a message as many times as it was suppressed in order to enable it. The **print_suppressed_messages** command displays currently suppressed messages.

EXAMPLES

When the argument to the **unalias** command does not match any existing aliases, the CMD-029 warning message displays. This example shows how to re-enable the suppressed CMD-029 message. Assume that there are no aliases beginning with 'q'.

```
prompt> unalias q*
prompt> unsuppress_message CMD-029
prompt> unalias q*
Warning: no aliases matched 'q*' (CMD-029)
```

SEE ALSO

print_suppressed_messages(2)
suppress_message(2)

update_block_views

This command detects any boundary change for readonly abstract blocks and incrementally updates the physical data for these blocks.

SYNTAX

int **update_block_views**
[-blocks *reference_names*]
[-verbose]

Data Types

reference_names list

ARGUMENTS

-blocks *reference_names*

Specifies a list of block references. When nothing is specified, command works on all the blocks.

-verbose

Specifies the verbosity of the command. By default verbose messaging is turned off.

DESCRIPTION

The **update_block_views** command updates the physical objects of a readonly abstract in case there is a boundary change for the abstract block. Prior to running this command, user is expected to reshape the block using GUI stretching utility.

This command works only on readonly abstract blocks.

This command will move/trim/remove the physical objects of that readonly abstract to honor the new block boundary. Additionally, this command will update locations information in the para file associated with the abstract. Also, new frame view will be created for this abstract. If there is an existing frame view, then new frame view will be created with same value of '-block_all' of the old abstract. If there isn't an existing frame view, then '-block_all true' will be used for frame creation. .

EXAMPLES

The following example updates the readonly abstract view for block reference "blk".

prompt> **update_block_views -blocks blk**

The following example checks for boundary changes for all the blocks and then updates them with verbosity turned on.

prompt> **update_block_views -verbose**

SEE ALSO

create_abstract(3)

update_constraint_mapping_file

Removes the specified constraint types for certain blocks in the mapping file.

SYNTAX

status **update_constraint_mapping_file**
[-remove_all]
[-remove_blocks *blocks*]
[-remove_types *constraint_types*]

Data Types

blocks collection
constraint_types collection

ARGUMENTS

-remove_all

Remove all constraint references for all blocks specified in the constraint mapping file. The **-remove_all** argument cannot be used together with the **-remove_blocks** or **-remove_types** arguments.

-remove_blocks *blocks*

Specifies a list of blocks for constraint removal, for all blocks specified in the constraint mapping file. All constraint mapping file references for the specified blocks are removed. If the **-remove_types** argument is not also specified, the tool removes all constraint mapping file references for all constraint types for the specified blocks. This argument is mutually exclusive with the **-remove_all** argument.

-remove_types *constraint_types*

Specifies a list of constraint types for removal. The *constraint_types* must be one or more of **DEF**, **UPF**, **ETM_UPF**, **SDC**, **BUDGET**, **CLKNET**, **PG_COSNTRAINT**, **COMPILE_PG**, **CTS_CONSTRAINT**, **BTM**, **FLOORPLAN** or **SCANDEF**. When no block is specified with the **-remove_blocks** argument, the tool removes the specified mapping file references for all blocks specified in the constraint mapping file. This argument is mutually exclusive with **-remove_all** argument.

DESCRIPTION

This command updates the current mapping between constraint mapping files and blocks. The current mapping is defined when you run the **set_constraint_mapping_file** command and specify a constraint mapping file. The constraint mapping file contains lines which specify a block name, constraint type, and corresponding constraint file name for each constraint. The supported

constraint types are DEF, UPF, ETM_UPF, SDC, BUDGET, CLKNET, PG_CONSTRAINT, COMPILE_PG, CTS_CONSTRAINT, BTM, FLOORPLAN and SCANDEF. After setting the constraint mapping file, the *update_constraint_mapping_file* command can be used to remove specified constraint type references for certain blocks, or remove all entries for all blocks in the map. Use the **report_constraint_mapping_file** command to verify the new mapping.

This command has three arguments: **-remove_all**, **-remove_types**, and **-remove_blocks**. At least one argument must be specified. To remove all constraint type entries for all blocks in the map, use the **-remove_all** argument. To remove all constraint map file entries for specified blocks, use the **-remove_blocks** argument. To remove the entries for specified types of constraints for all blocks in the map, use the **-remove_types** argument. To remove specific types of constraints for certain blocks, use both the **-remove_blocks** and **-remove_types** arguments.

EXAMPLES

The following constraint mapping file example specifies DEF, UPF and BTM constraint files for the blocks in the design. The files are used in the hierarchical flow from Verilog input to block shaping. The design has a top-level and two child blocks: BLK_A and BLK_B. The content of the constraint mapping file *./map_file.1* is:

```
BLK_A  DEF ./constraints/BLK_A.def
BLK_A  UPF ./constraints/BLK_A.upf
BLK_B  BTM ./constraints/BLK_B.btm
BLK_B  DEF ./constraints/BLK_B.def
BLK_B  UPF ./constraints/BLK_B.upf
TOP    DEF ./constraints/TOP.def
TOP    UPF ./constraints/TOP.upf
```

The following example sets the constraint mapping file using the preceding file, removes all constraint type entries for BLK_A, and reports the modified settings with the *report_constraint_mapping_file* command. Note that block BLK_A does not appear in output from the **report_constraint_mapping_file** command.

```
prompt> set_constraint_mapping_file ./map_file.1
prompt> update_constraint_mapping_file -remove_blocks BLK_A
prompt> report_constraint_mapping_file
BLK_B BTM ./constraints/BLK_B.btm
BLK_B DEF /disk1/constraints/BLK_B.def
BLK_B UPF /disk1/constraints/BLK_B.upf
TOP DEF /disk1/constraints/TOP.def
TOP UPF /disk1/constraints/TOP.upf
```

The following example removes the DEF constraint type for all blocks in the map. Note that constraint type DEF does not appear in output from the **report_constraint_mapping_file** command.

```
prompt> set_constraint_mapping_file ./map_file.1
prompt> update_constraint_mapping_file -remove_types DEF
prompt> report_constraint_mapping_file
BLK_A UPF ./constraints/BLK_A.upf
BLK_B BTM ./constraints/BLK_B.btm
BLK_B UPF /disk1/constraints/BLK_B.upf
TOP UPF /disk1/constraints/TOP.upf
```

The following example removes the UPF and BTM constraint types for block BLK_B and TOP.

```
prompt> set_constraint_mapping_file ./map_file.1
prompt> update_constraint_mapping_file -remove_types {UPF BTM} -remove_blocks {BLK_B TOP}
prompt> report_constraint_mapping_file
BLK_A DEF ./constraints/BLK_A.def
```

```
BLK_A UPF /disk1/constraints/BLK_A.upf
BLK_B DEF ./constraints/BLK_B.def
TOP DEF /disk1/constraints/TOP.def
```

SEE ALSO

report_constraint_mapping_file(2)
set_constraint_mapping_file(2)

update_cross_probing_files

Updates the paths of the cross-probed files.

SYNTAX

status **update_cross_probing_files**
[-search_path *paths*]
[-original_file *file1*]
[-new_file *file2*]
[-write_script_only *output_file*]
[-force]
[-verbose]

Data Types

paths list
output_file string
file1 string
file2 string

ARGUMENTS

-search_path *paths*

By default, the **search_path** variable is used to search for files when the tool updates the cross-probed files. If this option is specified, the tool uses the *paths* specified with this option instead.

When search paths are used, the first matching file found will be used as the replacement file. In general, a file is considered a match if its basename matches the original file and the file content is identical as determined by the tool.

-original_file *file1* / -new_file *file2*

Updates only one cross-probed file. The **-original_file** and **-new_file** options must be used together. The existing cross-probed file, *file1*, in the cross-probing database will be updated to the new file specified by *file2*. For the update to be successful,

- The *file1* argument needs to be an existing file in the set of cross-probed files as listed by the **report_cross_probing_files** command.
- The basename of the file in *file1* and *file2* need to match unless the **-force** option is used.
- The calculated checksum value of *file2* needs to match the one stored in the database for *file1*.

-write_script_only *output_file*

Specifies not to update the paths of the cross-probed files but writes the commands to a file. The file can be examined, modified if necessary, and then sourced as a Tcl script.

-force

Updates all cross-probed files (even the file paths are fine) if you specify this option without the **-original_file** and **-new_file** options. Without this option, the command updates files that do not have an OK status as reported by the **report_cross_probing_files** command.

When you specify the **-original_file** and **-new_file** options, the **-force** option allows the command to update the cross-probed file paths even if the basenames are different.

-verbose

Specifies to use verbose mode. The command prints out additional information messages.

DESCRIPTION

In order for GUI cross-probing functionality to work, the files that are stored in the database must be valid. If the design has been moved to a different location or cross-probed files have been moved to a different disk location, the paths that are stored in the database need to be updated accordingly. Running this command updates these paths.

To get a list of all cross-probed files and each file status, run the **report_cross_probing_files** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example uses the **-search_path** option with the **update_cross_probing_files** command:

```
prompt> report_cross_probing_files

****************************************
Report : cross_probing_files
Design : top
Version: ....
Date   : ....
****************************************

  Status:
          OK - No issues found
     Missing - File is missing
No Permissions - User has no read permissions
    Modified - File contents have been modified

Filename                    Status
------------------------------------------
/common/rtl/bot.v           Missing
/usrsite/joe/designs/mid.v  Missing
/usrsite/joe/designs/top.v  Modified

prompt> update_cross_probing_files -search_path /usrsite/common/rtl/
```

```
Original     /common/rtl/bot.v
Replacement  /usrsite/common/rtl/bot.v

Original     /usrsite/joe/designs/mid.v
Replacement  /usrsite/common/rtl/mid.v

Original     /usrsite/joe/designs/top.v
Replacement  /usrsite/common/rtl/top.v

prompt> report_cross_probing_files

****************************************
Report : cross_probing_files
Design : top
Version: ....
Date   : ....
****************************************

  Status:
          OK - No issues found
     Missing - File is missing
No Permissions - User has no read permissions
    Modified - File contents have been modified

Filename                    Status
------------------------------------------
/usrsite/common/rtl/bot.v       OK
/usrsite/common/rtl/mid.v       OK
/usrsite/common/rtl/top.v       OK
```

SEE ALSO

report_cross_probing_files(2)

update_timing

Updates timing information on the current design.

SYNTAX

string **update_timing**
[-full]

ARGUMENTS

-full

Indicates that the entire timing analysis is to be performed from the beginning. The default is to perform an incremental analysis, which updates only out-of-date information and runs more quickly.

DESCRIPTION

Updates timing for the current design. Timing is also automatically updated by commands that need the information, such as the **report_timing** command. This command explicitly prepares the design for further analysis.

By default, the **update_timing** command uses an efficient timing analysis algorithm that requires minimal computation effort and updates existing timing analysis information only where needed. You can override this default behavior using the **-full** option, which causes the entire timing update to be performed from the beginning. To avoid unnecessarily long run times, use the **-full** option only when you need to override incremental timing analysis. If the design is not timed, the **update_timing** command has the same runtime independent of the **-full** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example updates timing information for the current design.

prompt> **update_timing**

prompt> **update_timing -full**

SEE ALSO

report_timing(2)

update_topology_node

Updates multiple topology node attributes in a single command.

SYNTAX

collection **update_topology_node**
[*topology_node_list*]
[-name *topology_node_name*]
[-objects *object_list*]
[-constrained_objects *object_list*]
[-origin *coord*]
[-constraint_type edge | side | bbox | tap]
[-edge *edge_list*]
[-start *dist_or_pct*]
[-end *dist_or_pct*]
[-range *dist_or_pct*]
[-offset *distance*]
[-side *side_list*]
[-alignment left | right]
[-bbox_percentage *float_single_or_pair*]
[-corner_type auto | cross | default | river]
[-constraint_source user | application]
[-halo *distance*]

Data Types

topology_node_list collection
topology_node_name string
object_list collection
coord float_pair
edge_list integer_list
dist_or_pct float [+ "%"]
distance float
side_list string

ARGUMENTS

-name *topology_node_name*

Specifies the name of the topology_node. The name may not contain the special character '/'. If unspecified, the node's name will remain unchanged.

-objects *object_list*

Specifies the set of objects to be associated with the topology_node. The objects may be hierarchical. Allowed object types are pins, ports, cells, voltage_areas, voltage_area_shapes, shapes, placement_blockages, routing_blockages, bounds, and bound_shapes.

-constrained_objects *object_list*

Specifies the set of constrained objects to be associated with the topology_node. The objects may be hierarchical. Allowed object types are pins and ports.

-origin *coord*

Specifies the origin of the topology_node. If unspecified, the origin will be auto-computed based on the other specified options if possible.

-constraint_type edge | side | bbox | tap

Specifies the constraint type of the topology_node.

-edge *edge_list*

Specifies the constraint edge list of the topology_node. Value must be a list of integer values.

-start *dist_or_pct*

Specifies the constraint start of the topology_node. Value may be either a distance or a percentage - to specify a percentage, append a '%' character to the end of the float value.

-end *dist_or_pct*

Specifies the constraint end of the topology_node. Value may be either a distance or a percentage - to specify a percentage, append a '%' character to the end of the float value.

-range *dist_or_pct*

Specifies the constraint range of the topology_node. Value may be either a distance or a percentage - to specify a percentage, append a '%' character to the end of the float value.

-offset *distance*

Specifies the constraint offset of the topolology_node.

-side *side_list*

Specifies the constraint sides of the topolology_node. Value may be a single instance or list of the following: W, N, E, or S.

-alignment left | right

Specifies the constraint alignment of the topolology_node.

-bbox_percentage *float_single_or_pair*

Specifies the bbox_percentage of the topology_node. Specify a single value for linear regions, and a float pair for 2D regions.

-corner_type auto | cross | default | river

Specifies the corner_type of the topology_node.

-constraint_source user | application

Specifies the constraint_source of the topology_node.

-halo *distance*

Specifies the halo of the topology_node. May only be specified when the *-origin* argument is specified.

DESCRIPTION

The **update_topology_node** command allows the user to modify several topology_node attributes at once, instead of incrementally using the *set_attribute* command.

EXAMPLES

The following example sets a topology_node to constraint_type "origin":

```
prompt> update_topology_node [get_topology_nodes TOPOLOGY_NODE0] -constraint_type origin -origin {1.0 2.4}
{curplan/TOPOLOGY_NODE0}
```

SEE ALSO

create_topology_edge(2)
current_topology_plan(2)
get_topology_edges(2)
get_topology_nodes(2)
remove_topology_edges(2)
remove_topology_nodes(2)
report_topology_plans(2)

upf_version

Displays the version of UPF currently being used to interpret UPF commands.

SYNTAX

string **upf_version**
[*version*]

Data Types

version String

ARGUMENTS

version

Documents the UPF version for which the UPF commands that follow were written.

DESCRIPTION

This command returns a string value representing the version of UPF used by the tool to interpret other UPF commands.

An optional **version** argument can be given to the command; it will be ignored and won't have any impact on the syntax or semantics of any of the UPF commands, as its only purpose is to document the intended version that should be used to interpret the commands that follow.

Any instance of usage will be preserved as part of the UPF file that can be retrieved using the **save_upf** command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example makes use of the command using the optional argument.

```
prompt> upf_version 2.1
IEEE-1801
```

SEE ALSO

save_upf(2)

use_interface_cell

Specifies how to map the isolation, level-shifter, and enable level-shifter cells belonging to the specified isolation and/or level-shifter strategy.

SYNTAX

status **use_interface_cell**
interface_implementation_name
-strategy *list_of_isolation_level_shifter_strategies*
-domain *power_domain*
-lib_cells *lib_cells*

Data Types

interface_implementation_name string
list_of_isolation_level_shifter_strategies list
power_domain string
lib_cells list

ARGUMENTS

interface_implementation_name

Specify the name of the interface cell implementation strategy.

list_of_isolation_level_shifter_strategies

Specify one of the following: - one isolation strategy, or - one level shifter strategy, or - a pair of isolation and level shifter strategies No more than one isolation or level shifter strategy is allowed in one command.

-domain *power_domain*

Specifies the name of the power domain name to which the strategy/strategies belongs.

-lib_cells *lib_cells*

Specifies a list of the target library cells to be used for the isolation mapping.

DESCRIPTION

This command defines how the **compile** command performs the mapping of isolation, level-shifter, or enable level-shifter cells.

EXAMPLES

In the following example, the isolation cells belonging to the strategy named ISO1 will be mapped to the isocell1 library cell with the highest priority, and level-shifter cells belonging to the strategy LS1 will be mapped to the levshicell1 library cell with the highest priority:

```
prompt> use_interface_cell my_interface -domain PD1 -strategy {ISO1 LS1} \
 -lib_cells {isocell1 levshicell1} \
 -port_map {{VDD sn1} {VSS gnd}}
```

SEE ALSO

set_isolation(2)
set_level_shifter(2)

vclp_stop

Used to terminate vclp if vclp was invoked by the user by running check_vclp_design.

SYNTAX

status **vclp_stop**

DESCRIPTION

The **vclp_stop** command terminates vclp, if vclp was invoked by the user within this tool.

SEE ALSO

check_vclp_design(2)

vclp_zoom_highlight

This command is used to highlight the instance that is selected in verdi GUI in ICC2/FC GUI.

SYNTAX

status **vclp_zoom_highlight**

DESCRIPTION

If users have invoked VCLP with **check_vclp_design** and are interacting with the Verdi GUI, they can use the **vclp_zoom_highlight** command to highlight the instance that was selected in Verdi GUI in ICC2/FC GUI.

SEE ALSO

check_vclp_design(2)

verify_via_ladders

Verifies the via ladders in a design to confirm that the via ladders match user-specified via ladder constraints and that the via ladders are properly connected to pins.

SYNTAX

status **verify_via_ladders**
[-nets {*collection_of_nets*}]
[-shift_vias_on_transition_layers true | false]
[-report_all_via_ladders true | false]

Data Types

{*collection_of_nets*} collection

ARGUMENTS

-nets {*collection_of_nets*}

Specifies the nets on which to verify via ladders.

Via ladders will be verified only on the specified nets.

If you do not specify this option, all via ladders will be verified.

-shift_vias_on_transition_layers true | false

Specifies whether the command should expect that via ladder cuts on transition layers have been shifted off the routing tracks.

The default is false.

If via ladders were inserted using insert_via_ladders -shift_vias_on_transition_layers true, the via ladders should be verified using verify_via_ladders -shift_vias_on_transition_layers true.

If via ladders were inserted using insert_via_ladders -shift_vias_on_transition_layers false, the via ladders should be verified using verify_via_ladders -shift_vias_on_transition_layers false.

If false (the default), via ladder cuts must be centered at the intersection of routing tracks to be considered correctly placed.

If true, via ladder cuts are allowed to be shifted off track on transition layers.

-report_all_via_ladders true | false

Species whether to print all the entries of via ladder verification report.

The default is true.

If false, the report only prints the first 40 entries for each category.

DESCRIPTION

The **verify_via_ladders** command verifies the via ladders in the layout of a design.

This command does not perform conventional design rule checking on via ladders. The command checks that via ladders in the design are properly connected to pins and that the via ladders in the layout match the via ladder constraints on the corresponding pins.

The command reports all via ladders in the layout, and the matching tech file ViaRule template. A "Misplaced via ladder" DRC is reported if a via ladder is dangling or floating or not properly connected to a pin. A "Misplaced via ladder" DRC is reported if a via ladder is connected to a pin but does not match the via ladder constraints for that pin. A "Needs via ladder" DRC is reported for any pin that has a via ladder constraint but does not have a matching physical via ladder in the layout.

EXAMPLES

The following example verifies all via ladders in the design.

```
prompt> verify_via_ladders
```

The following example will verify all via ladders in the design and assumes that these were inserted using insert_via_ladders -shift_vias_on_transition_layers true.

```
prompt> verify_via_ladders -shift_vias_on_transition_layers true
```

SEE ALSO

set_via_ladder_constraints(2)
remove_via_ladder_constraints(2)
remove_via_ladders(2)
insert_via_ladders(2)
refresh_via_ladders(2)
set_via_ladder_rules(2)

which

Locates a file and displays its pathname.

SYNTAX

string **which** *filename_list*

list *filename_list*

ARGUMENTS

filename_list

List of files to locate.

DESCRIPTION

Displays the location of the specified files. This command uses the search_path to find the location of the files. This command can be a useful prelude to read_db or link_design, because it shows how these commands expand filenames. The **which** command can be used to verify that a file exists in the system.

If an absolute pathname is given, the command searches for the file in the given path and returns the full pathname of the file.

EXAMPLES

The following examples are based on the following search_path.

prompt> **set search_path "/u/foo /u/foo/test"**

The following command searches for the file name foo1 in the search_path.

prompt> **which foo1**
/u/foo/foo1

The following command searches for files foo2, foo3.

prompt> **which {foo2 foo3}**
/u/foo/test/foo2 /u/foo/test/foo3

The following command returns the full pathname.

```
prompt> which ~/test/designs/sub_design.db
/u/foo/test/designs/sub_design.db
```

SEE ALSO

link_design(2)
read_db(2)
search_path(3)

widen_wires

Performs wire widening on the current design and reduces the open critical area to improve the yield.

SYNTAX

status **widen_wires**
[-spreading_widening_relative_weight *ratio_value*]
[-widen_widths_by_layer_name {*list_of_layer_value_lists*}]
[-timing_preserve_nets {*collection_of_nets*}]
[-timing_preserve_setup_slack_threshold *slack_value*]
[-timing_preserve_hold_slack_threshold *slack_value*]

Data Types

count integer
collection_of_nets collection
slack_value float
ratio_value float
list_of_layer_value_lists collection

ARGUMENTS

-spreading_widening_relative_weight *ratio_value*

Controls the relative preference for wire spreading versus wire widening.

During wire widening, Zroute increases the wire width to reduce open critical area. Due to the limited routing resources, this decreases the spacing to the neighboring wires and can reduce the improvement in critical area shorts that occurred during wire spreading.

If there are conflicts between the desired width and the desired spacing, you can use this option to alter the weight on either spacing or width according to your preference. If the value is set to 0.5, it means spacing and width have equal weight. If the value is less than 0.5, more weight is put on width. If the value is greater than 0.5, more weight is put on spacing. The greater the weight on width, the greater the expected improvement for open critical area. The greater the weight on spacing, the greater the expected improvement for critical area shorts.

You can specify an floating-point number between 0.0 and 1.0. The default is 0.5.

-widen_widths_by_layer_name {*list_of_layer_value_lists*}

Specifies a list of desired wire widths on each layer for the router to try. You can specify a maximum of five wire widths for each layer. You must specify the layers by using the layer names in the technology file.

Use the following format to specify the allowed wire widths for each layer:

{*layerName value1* [*value2* [*value3* [*value4* [*value5*]]]]}

For example, {{metal2 0.07 0.06} {metal3 0.08}}.

The wire width values should be between the default wire width and the maximum width that does not trigger any fat wire rules. The tool validates the input wire widths and skips them if they are not valid. If none of the wire widths specified for a layer is valid, the default wire width values will be used for such layer.

The default value for the maximum wire width for a layer is about 1.5 times the layer default width.

-timing_preserve_nets {*collection_of_nets*}

Specifies the nets to preserve timing on.

If you specify this option, the command considers the specified nets as timing critical and skips wire widening on these nets and the neighboring nets.

-timing_preserve_setup_slack_threshold *slack_value*

Specifies the setup slack threshold for timing critical nets.

If you specify this option, the command skips wire widening on all nets with a worst setup slack less than or equal to this value and the neighboring nets.

-timing_preserve_hold_slack_threshold *slack_value*

Specifies the hold slack threshold for timing critical nets.

If you specify this option, the command skips wire widening on all nets with a worst hold slack less than or equal to this value and the neighboring nets.

DESCRIPTION

The **widen_wires** command performs wire widening to reduce the total open critical area. It attempts to widen every wire in the design. The options control the maximum number of search and repair iterations performed after wire widening to resolve any violations.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following command performs wire widening.

```
prompt> widen_wires
```

SEE ALSO

route_detail(2)
spread_wires(2)

win_select_objects

Creates a collection of objects equivalent to a graphical selection operation.

SYNTAX

string **win_select_objects**
[-slct_targets *slct_bus*]
[-slct_targets_operation *operation*]
[-create_slct_buses]
[-root *instance*]
[-within *rectangle* | -line *line* |-at *point* |
-radius *r* | -again_at]
[-intersect]
[-index *i*]
[-visible]

string *slct_bus*
string *operation*
string *instance*

ARGUMENTS

-slct_targets *slct_bus*

Specifies the name of a selection bus to store the result in. By default the result of this command is returned in a collection. If this option is specified selection bus *slct_bus* is used instead. *slct_bus* is also returned as result of the command.

-slct_targets_operation *operation*

Specifies which operation should be used when storing the result in selection bus *slct_bus*. This is only allowed if you specify the -slct_targets option. Legal operations are clear, add and remove.

-create_slct_buses

Specifies that a new selection bus is created and used to store the result in. Using this option is equivalent to first creating a selection bus using the create_selection_bus command and then providing the created selection bus to option -slct_targets.

-root *instance*

Specifies a string for the instance whose component design objects and hierarchy are to be examined against the specified region criteria.

-within "*rectangle*"

Collects design objects lying within the specified rectangle.

-line "*line*"

Collects design objects intersected by specified line.

-at "*point*"

Collects design objects whose bounding box contains the specified point.

-radius *r*

Specifies a radius for points specified by the **-at** option. It is applicable only for **_pins** and **_ports**.

-again_at

Reapplies the previous **-at** option, but returns a previously-matched design object.

-index *i*

For point select (-at option) specifies a specific object index to select. Similar to running -again_at.

-intersect

Collects design objects intersecting with specified rectangle. Without this option, the command collects design objects contained in the specified rectangle.

-visible

Select objects marked as visible with win_set_select_class command.

DESCRIPTION

This command is for use by the graphics window(s) for logging interactive, graphical selection operations to the command log for later playback. You are advised not to use this command directly. Use of this command other than by graphics windows may cause unexpected results from selection operations.

SEE ALSO

win_set_select_class(2)
win_set_filter(2)

win_set_filter

Sets a filter to apply to objects selected by the **win_select_objects** command.

SYNTAX

int **win_set_filter**
-class *class_name*
[-stop_level *level*]
[-start_level *level*]
[-z_level *level*]
[-filter *expression*]
[-layer *list*]
[-user_filter *true*|*false*]
[-user_filter_cmd *tcl_cmd*]
[-highlighted_only *true*|*false*]
[-expand_cell_types *list*]
[-visible]

ARGUMENTS

-class "*class_name*"

Specifies a single class name for which the filter is to apply.

-stop_level "*level*"

Specifies the hierarchy level, up to which **win_select_objects** searches for design objects down the hierarchy. This has no effect on objects that are not collectable.

-start_level "*level*"

Specifies the hierarchy level, from which **win_select_objects** begins searching for design objects down the hierarchy. This has no effect on objects that are not collectable.

-z_level "*level*"

Specifies the interesting Z levels in a 3dic design. The 0 value means all chips in the current 3dic design. Other values mean only the chips on this level and the level below. For example 1 means chips on level 0 and level 1.

-filter "*expression*"

Specifies a filter expression that an object must satisfy to be returned by the **win_select_objects** command. The expression argument must be a Boolean expression based on attributes of the specified class. If this option is not specified, the filter is cleared.

-layer "*list*"

Specifies a layer number list that an object must satisfy to be returned by the **win_select_objects** command. If this option is not specified, the layer filter is cleared.

-user_filter "*true*|*false*"

Enables user supplied filter command.

-user_filter_cmd "*tcl_cmd*"

Specifies a tcl command for filtering.

-highlighted_only "*true*|*false*"

Specifies that only highlighted objects can be selected.

-expand_cell_types "*list*"

Specifies cell types for searching design objects down the hierarchy when the search level is greater than 0. This has no effect on types that is not supported or objects that are not collectable.

-visible

Specified filters are for visible objects. By default the filters are for selectable objects.

DESCRIPTION

This command is for use by the graphics window(s) for logging interactive, graphical selection operations to the command log for later playback. You are advised not to use this command directly. Use of this command other than by graphics windows may cause unexpected results from selection operations.

SEE ALSO

win_select_objects(2)
win_set_select_class(2)

win_set_select_class

Sets the design objects to be collected by the **win_select_objects** command.

SYNTAX

string **win_set_select_class**
{-all | *class_names*}
[-visible]

ARGUMENTS

-all

Selects all classes. The **-all** option and the *class_names* option are mutually exclusive.

class_names

Specifies a list of design objects. The **-all** option and the *class_names* option are mutually exclusive.

-visible

Specified classes are for visible objects. By default the classes are for selectable objects.

DESCRIPTION

This command is for use by the graphics window(s) for logging interactive, graphical selection operations to the command log for later playback. You are advised not to use this command directly. Use of this command other than by graphics windows may cause unexpected results from selection operations.

SEE ALSO

win_select_objects(2)
win_set_filter(2)

win_snap_point

Snaps a point to a grid or to the nearest object. The result of this command is returned as a point. If the point cannot be snapped the result point is equal to the given point.

SYNTAX

string **win_snap_point**
-point *point*
[-snap_type *snap_type*]
[-radius *r*]

string *snap_type*

ARGUMENTS

-snap_type *snap_type*

Specifies the type of snapping to use. It is one of these values [litho (default), site, wiretrack, user, objects]. In case of snapping to the nereast objects, the objects will be chosen based on filters set by previous calls of *win_set_select_class -visible* and *win_set_filter -visible* commands.

-point "*point*"

Specifies a point to snap.

-radius *r*

Specifies a radius for point specified by the **-point** option. It is applicable only for snapping to objects.

DESCRIPTION

This command is for use by the graphics window(s) while editing. You are advised not to use this command directly.

EXAMPLES

icc_shell> **win_snap_point -snap_type litho -point {123.456 456.123}**
123.460 456.120

SEE ALSO

win_set_select_class(2)
win_set_filter(2)

write_aif

Writes bump locations and connected nets or ports to an Advanced Input Format (AIF) file.

SYNTAX

status **write_aif**
[-use_port_name]
[-hierarchy]
[-bumps *bump_cell_collection*]
aif_file_name

Data Types

bump_cell_collection collection
aif_file_name string
bump_objects_collection collection

ARGUMENTS

-use_port_name

Writes the associated port name for each bump to the AIF file. If a bump does not have an associated port, the command writes the associated net name.

By default, the command writes the associated net name for each bump.

-hierarchy

Writes information for bumps within child cells of the current block. If a bump connects to a port or net within the hierarchy, use its top-level port or net and top-level location.

When you use this option, the generated AIF file contains the following note at the beginning of the file:

;It was written by write_aif with option -hierarchy

By default, only the top-level bump instances are reported in the AIF file.

-bumps *bump_cell_collection*

Specifies the bump instances to write to the AIF file.

This option by itself specifies bump instances at the top level. If you include bump instances in child cells, you must also specify the **-hierarchy** option.

By default, all top-level bump instances are written to the AIF file.

-bump_objects *bump_objects_collection*

Specifies the bump pins or pin shapes to write to the AIF file in NETLIST section. You can specify a mixture of pin and pin shapes

This option by itself specifies bump pins or pin shapes that are not in bump cells. These pins or pin shapes are not written to the AIF file by default.

aif_file_name

Specifies the name of the generated AIF file.

DESCRIPTION

The **write_aif** command writes information about the specified bump instances to the specified AIF file. You can specify the file with a relative or absolute path.

The AIF file includes the following information for the bump instances:

- Connectivity

  By default, the connectivity is the name of the net connected to the bump instance. To use the connected port name instead, use the **-use_port_name** option.

- Reference bump cell
- Orientation
- Location

  The location represents the center of the bump instance. The coordinates of the bump center reference the center of the die, which is defined as the center of die area object in the block and are specified in the [DIE] section of the AIF file. The command uses the following formula to calculate the coordinates of the center of the bump instance:

  ```
  Pad_center_x = AIF_Pad_x + die_width/2 + die_offset_x
  Pad_center_y = AIF_Pad_y + die_height/2 + die_offset_y
  ```

The following is an example AIF file.

```
;This is a comment

[DATABASE]
TYPE=AIF
VERSION=2.0
UNITS=UM

[DIE]
WIDTH=3230.000000
HEIGHT=2181.600000

[PADS]

PAD80B_N = POLY 16.570,40.000  40.000,16.570  40.000,-16.570  16.570,-40.000
      -16.570,-40.000  -40.000,-16.570  -40.000,16.570  -16.570,40.000
      16.570,40.000
PAD80B_E = POLY 16.570,40.000  40.000,16.570  40.000,-16.570  16.570,-40.000
      -16.570,-40.000  -40.000,-16.570  -40.000,16.570  -16.570,40.000
      16.570,40.000
PAD80B_S = POLY 16.570,40.000  40.000,16.570  40.000,-16.570  16.570,-40.000
```

```
        -16.570,-40.000  -40.000,-16.570  -40.000,16.570  -16.570,40.000
        16.570,40.000
PAD80B_W = POLY 16.570,40.000  40.000,16.570  40.000,-16.570  16.570,-40.000
        -16.570,-40.000  -40.000,-16.570  -40.000,16.570  -16.570,40.000
        16.570,40.000
PAD80B_P_N = POLY 16.570,40.000  40.000,16.570  40.000,-16.570  16.570,-40.000
        -16.570,-40.000  -40.000,-16.570  -40.000,16.570  -16.570,40.000
        16.570,40.000
PAD80B_G_N = POLY 16.570,40.000  40.000,16.570  40.000,-16.570  16.570,-40.000
        -16.570,-40.000  -40.000,-16.570  -40.000,16.570  -16.570,40.000
        16.570,40.000

[NETLIST]

;Netname Pad#    Type       Pad_X     Pad_Y     Ball#

-     BUMP_0   PAD80B_N    -1425.0000  -900.8000
-     BUMP_1   PAD80B_E    -1425.0000  -750.8000
-     BUMP_2   PAD80B_N    -1425.0000  -600.8000
-     BUMP_3   PAD80B_N    -1425.0000  -450.8000
NET_0   BUMP_4   PAD80B_S    -1425.0000  -300.8000
NET_1   BUMP_5   PAD80B_N    -1425.0000  -150.8000
NET_2   BUMP_6   PAD80B_N    -1425.0000  -0.8000
-     BUMP_7   PAD80B_W    -1425.0000  149.2000
NET_3   BUMP_8   PAD80B_P_N   -1425.0000  299.2000
NET_4   BUMP_9   PAD80B_G_N   -1425.0000  449.2000
NET_5   BUMP_10  PAD80B_N    -1425.0000  599.2000
```

The following list describes the file sections in the AIF file:

- [DATABASE]

  Identifies the file as an AIF file and contains the version and units of the coordinates.

- [DIE]

  Describes the die width, height, and die name. Width is measured along the horizontal x-axis and height is measured along the vertical y-axis. The width and height are defined per the die nominally described dimensions.

- [PADS]

  The PADS section defines all pad types needed for the die.

- [NETLIST]

  Defines each net and the coordinates of each die pad. Each column requires an entry. If a particular data item is not present, you must use a hyphen as a placeholder.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples writes all top-level bump instances to an AIF file named my.aif.

```
prompt> write_aif my.aif
1
```

The following example writes the specified bump instance to an AIF file named my.aif. The connectivity column of the AIF file reports the ports connected to the bump instances.

```
prompt> write_aif my.aif -use_port_name -bumps [get_cells iBump_*]
1
```

SEE ALSO

read_aif(2)

write_app_options

Writes application option data to a binary data file for later reporting by the **report_app_options** command and comparison by the **compare_app_options** command.

SYNTAX

integer **write_app_options**
-output *data_file*
[-non_default]

Data Types

data_file string

ARGUMENTS

-output *data_file*

Specifies the output file name to contain the application option settings and metadata. The *data_file* argument can be a local file name or a full path name to the file.

-non_default

Writes out data only for application options with nondefault values.

DESCRIPTION

This command generates an encoded binary data file that contains information on all application options and their current values and metadata.

If application option values were set with the **set_app_options** command, the source location of the call to **set_app_options** will be recorded.

If an internal application options has a non-default value, the tool issues the following message:

"There are additional internal differences with no available TBC source."

Or...

"There are additional internal differences."

This message indicates that the design contains Synopsys internal application options that were possibly set from a .tbc file or by

using some other method.

The **-non_default** option specifies that only application options with nondefault values are written to the data file. Otherwise, all application option values are written to the data file.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example creates a encoded binary data file that contains all application options with non-default values. Use the **report_app_options** and **compare_app_options** commands to examine the data.

prompt> **write_app_options -output app_option.data -non_default**

The following example creates a encoded binary data file that contains all application options. The **-non_default** option is omitted to allow a comprehensive comparison between two tool sessions for all application options using this data, including application option withs default values, allowed values, block scope vs. global scope, and so on.

prompt> **write_app_options -output SP1.data**

SEE ALSO

compare_app_options(2)
get_app_options(2)
help_app_options(2)
report_app_options(2)
reset_app_options(2)
set_app_options(2)

write_app_var

Writes a script to set the current variable values.

SYNTAX

string **write_app_var**
-output *file*
[-all | -only_changed_vars]
[*pattern*]

Data Types

file string
pattern string

ARGUMENTS

-output *file*

Specifies the file to which to write the script.

-all

Writes the default values in addition to the current values of the variables.

-only_changed_vars

Writes only the changed variables. This is the default when no options are specified.

pattern

Writes the variables that match the specified *pattern*. The default is "*".

DESCRIPTION

The **write_app_var** command generates a Tcl script to set all application variables to their current values. By default, variables set to their default values are not included in the script. You can force the default values to be included by specifying the **-all** option.

EXAMPLES

The following is an example of the **write_app_var** command:

```
prompt> write_app_var -output sh_settings.tcl sh*
```

SEE ALSO

get_app_var(2)
report_app_var(2)
set_app_var(2)

write_ascii_files

Writes the design data and shell commands for the current design so that the settings are saved when you load the design back into the tool.

SYNTAX

status **write_ascii_files**
-output *dir_name*
[-modes *mode_list*]
[-corners *corner_list*]
[-include *include_list*]
[-exclude *exclude_list*]
[-golden_upf_file *upf_file*]
[-golden_floorplan]
[-no_library_creation]
[-multicore]
[-force]
[-def_units 100 | 200 | 1000 | 2000 | 10000 | 20000]
[-def_version *version*]
[-compress gzip]

Data Types

mode_list	list
corner_list	list
include_list	list
exclude_list	list
dir_name	string
upf_file	string

ARGUMENTS

-modes *mode_list*

Specifies the modes to write.

By default, all modes are written.

-corners *corner_list*

Specifies the corners to write.

By default, all corners are written.

-include *include_list*

Specifies a list of constructs and objects to include in the output. When this option is used, the Verilog netlist contains the basic logic netlist and only the types of constructs and objects specified with this option. The valid values are:

all
all_physical_cells
analog_pg
corner_cells
cover_cells
diode_cells
empty_modules
end_cap_cells
feedthrough_cells
filler_cells
flip_chip_driver_cells
flip_chip_pad_cells
leaf_module_declarations
pad_cells
pad_spacer_cells
pg_netlist
pg_objects
physical_only_cells
scalar_wire_declarations
shadow_netlist
spare_cells
supply_statements
unconnected_ports
user_pg
well_tap_cells

For a description of each construct and object type, see the DESCRIPTION section. The default behavior is to include everything except for scalar wire declarations, leaf module declarations, PG objects, and connections to unconnected ports.

-exclude *exclude_list*

Specifies a list of constructs and objects to exclude from the output. When this option is used, the Verilog netlist contains everything except for the types of constructs and objects specified with this option.

The valid values are:

all_physical_cells
analog_pg
corner_cells
cover_cells
diode_cells
empty_modules
end_cap_cells
feedthrough_cells
filler_cells
flip_chip_driver_cells
flip_chip_pad_cells
leaf_module_declarations
pad_cells
pad_spacer_cells
pg_netlist
pg_objects
physical_only_cells
scalar_wire_declarations
shadow_netlist

spare_cells
supply_statements
unconnected_ports
well_tap_cells

-output *dir_name*

Specifies the name of the directory in which to write the script files.

-no_library_creation

Exclude library creation and library setup

-multicore

Enable multicore support

-force

Overwrites the existing script files. By default, the command fails if the script file already exists.

-golden_floorplan

Specifies to use the golden floorplan.

-golden_upf_file *upf_file*

Specifies the golden UPF file.

-def_units 100 | 200 | 1000 | 2000 | 10000 | 20000

Specifies the number of units per micron in the DEF file. All database length units are scaled accordingly in the output DEF. If units are not specified, the units per micron is obtained from the technology length precision. Valid values are 100, 200, 1000, 2000, 10000, and 20000. This option is passed to the DEF writer and the behavior is identical to the **-units** option of the *write_def* command.

-def_version *version*

Specifies the DEF syntax version used to create floorplan.def file. Valid values are 5.7 and 5.8. The default version is 5.7.

-compress gzip

Specifies the method used to compress the generated script files.

By default, the command writes the script files as simple text files and does not compress them.

DESCRIPTION

The **write_ascii_files** command writes the design files and Tcl scripts for the current design to load the design data into back into the tool.

Files are written into the directory specified by the -output option. The written files include the following, if the data is present:

Netlist in Verilog format
SAIF name mapping
Tcl floorplan and DEF files
Design's power intent as a UPF command script
Scan chain information in SCANDEF format

Constraints in SDC format
Timing contexts (scenarios)
Cell expansion data
RP groups

In addition, the write_ascii_files command writes the following:

Tool application option settings
The top-level script to load the generated data back into the tool
Library setup information

The write_ascii_files command saves the design data for the current design and names the output files with the design name. The top-level script, **<design_name>**.fc_script.tcl, loads all the design data in the correct order. By default the tool includes a script, **<design_name>**.library_setup.tcl, to create a design library. In the golden UPF mode, the command writes out the supplemental UPF file and the name mapping file, and the top-level script includes load_upf commands to load the golden files specified by the -golden_upf option.

The top-level script does not include the commit_upf or update_timing commands after loading the design.

The -include and -exclude options are passed directly to the write_verilog command. For more information about the behavior of this option please review the write_verilog man page.

The **write_ascii_files** command includes an execution log file that can be reviewed to check the messages issued by the tool when writing out the various files.

Multicorner-Multimode Support

By default, this command uses information from all modes and all corners.

You can specify the modes and corners to write by using the **-modes** and **-corners** options.

EXAMPLES

The following example writes the settings on the current design to the test_scr directory.

```
prompt> write_ascii_files -output test_scr
```

SEE ALSO

change_names
write_verilog
write_script
save_upf
saif_map
write_cell_expansion
write_scan_def
write_rp_groups
write_floorplan
commit_upf
update_timing

write_blackbox_timing_script

Writes a script that contains the black box timing information for the design.

SYNTAX

status **write_blackbox_timing_script**
output_file_name | -qtm_format [-qtm_directory *directory_name*]

Data Types

output_file_name string
directory_name string

ARGUMENTS

output_file_name

Specifies the name of the black box timing file to write to. This option is mutually exclusive with **-qtm_format** and you must specify one of them.

-qtm_format

Writes out black box timing data in Quick Timing Model (QTM) format. This option is mutually exclusive with *output_file_name* and you must specify one of them.

-qtm_directory *directory_name*

Specifies the directory in which to write the QTM files. The tool writes out one QTM file for each scenario. Depending on the number of scenarios in the design, the tool might need to write out more than one QTM file. The default directory is *bbt_qtm_dir*. This option can only be specified together with **-qtm_format**.

DESCRIPTION

This command writes out black box timing information to either a file in black box timing (BBT) format or a set of files in QTM format. If it writes out a file in BBT format, the file contains the commands to regenerate the black box timing data for the current design in this tool. If it writes out one or more QTM files, the files would contain QTM commands and can be used to generate .db files to model the black box design's timing. These files can also be used in the PrimeTime tool.

The design must already have black box timing defined.

EXAMPLES

The following example sources a file that creates black box timing for the current design, then writes out a script that contains the defined black box timing.

```
prompt> source block.bbt.tcl
prompt> write_blackbox_timing_script bbt_new.tcl
```

The following example sources a file that creates black box timing for the current design, then writes out QTM for the defined black box timing.

```
prompt> source block.bbt.tcl
prompt> write_blackbox_timing_script -qtm_format -qtm_directory qtm_files
```

SEE ALSO

commit_blackbox_timing(2)

write_budgets

Writes SDC I/O constraints for budgeted blocks to disk.

SYNTAX

int **write_budgets**
[-blocks *block_cells*]
[-top]
[-full_budget_blocks *block_module_names*]
[-design_subblocks *block_module_names*]
[-hier_abstract_subblocks *block_module_names*]
[-shell_subblocks *block_module_names*]
[-output *dir_name*]
[-force]
[-verbose]
[-nosplit]
[-compress gzip]
[-format dc | icc2]

Data Types

block_cells collection
block_module_names collection
dir_name string

ARGUMENTS

-blocks *block_cells*

Writes block budgets for only the named cell instances. It is an error to include instances which were not previously declared with the **-add_blocks** option of the **set_budget_options** command.

-top

Writes a budget for the top level.

-full_budget_blocks *block_module_names*

Writes "full" budgets for the specified blocks, instead of the default incremental budget. See the section on "Budget Constraint Styles" for more information.

You can set the **plan.budget.all_full_budgets** application option to true to automatically set this option for all blocks. The tool always writes a full budget for the top module.

-design_subblocks *block_module_names*

Keeps internal constraints for subblocks in the budget. Note that this option has no effect unless you also select the **-full_budget_blocks** option for the parent of the subblock. Please refer to the man page for **split_constraints** for a full description of this option. You can use the **plan.budget.all_design_subblocks** application option to automatically set this option for all blocks.

-hier_abstract_subblocks ***block_module_names***

Retain some extra internal constraints of subblocks in the budget. Note that this option has no effect unless you also select the **-full_budget_blocks** option for the parent of the subblock. Please refer to the man page for **split_constraints** for a full description of this option.

-shell_subblocks ***block_module_names***

Creates timing budgets as a "budget shell" for each specified block. When you use a budget shell, the contexts of your block is ignored, and the boundary timing of your block is based only on your budget constraints. Note that budget shells require you to relink your design using shell libraries instead of a netlist or an abstract. See "hierarchical budgeting" below for details.

Note that using **-shell_subblocks** automatic adds the parent of the subblock to the list of blocks for **-full_budget_blocks**. Parent blocks of shell blocks cannot have incremental budgets.

-output ***dir_name***

Specifies the name of the directory under which the scripts are written. By default, the directory is "./budgets". For each budgeted block, a subdirectory is added under your specified directory. The subdirectory has the same name as the block.

-force

Forces the output directory to be overwritten, if it already exists.

-verbose

Prints detailed warning messages about pins with missing budget specifications. By default, only a summary is printed. You can use the **report_budget** command to show the verbose feedback at any time.

-nosplit

Writes out long constraint command lines, even if the line length is greater than 80 columns. Use this option to simplify post-processing of the output constraint files or when comparing the constraint files by using the UNIX **diff** command. If this option is not specified, the command breaks long constraint command lines near column 80 and inserts a line continuation character (\).

-compress gzip

Specifies the method used to compress the generated script files. By default, the command writes the script files as simple text files and does not compress them.

-format dc | icc2

Specifies the tool in which the budget shells created by the command will be used.

Valid values are:

- icc2 (the default)
  The budget shells generated can be used by ICC2
- dc
  Besides the budget shells db files, other files are created to facilitate the ICC2->DC flow. The tool outputs timing constraints that can be read in Design Compiler as well as Milkyway libraries to be used as reference libraries in DC.

This option can only be used together with the **-shell_subblocks** option.

DESCRIPTION

This command writes SDC constraints for your budgets to disk. For each budgeted subblock, a separate subdirectory is written under the directory specified by the **-output** option. The name of the subdirectory will correspond to the module name of a budgeted block. The command writes multiple files within each subdirectory, similar to those created by the **write_script** command.

You are responsible for applying the budget constraints to your lower-level blocks. This is done by sourcing the file called "top.tcl" from the **write_budgets** output. For example, if your module name is MY_BLOCK, and you have written your budgets to a directory called "budgets", you should use the following command to apply budget constraints to your lower-level blocks:

```
source budgets/MY_BLOCK/top.tcl
```

The **source top.tcl** command loads other files corresponding to constraints for block modes and corners. The number of mode and corner files written depends on what modes and corners are currently defined at the top level, and on any mode and corner settings you declared with the **set_block_to_top_map** command.

Using write_budgets in the Flow

In a typical design flow, the lowest level blocks in your design are represented by block abstracts. Block abstracts save significant memory and CPU. When processing block abstracts, the **write_budgets** command only writes constraints that apply to the boundary of a subblock. The output of **write_budgets** is *not* a complete set of constraints for a block. Rather, the output is intended to be "layered" incrementally on top of your existing block constraints. To use the budget file from **write_budgets**, you need to perform the following steps:

- Load your block
- Apply your regular block constraints that constrain the internal parts of your block.
- At the top level, obtain budget constraints automatically by using **compute_budget_constraints** or manually using**set_pin_budget_constraints** and other related commands. If you want to generate hold constraints as well as setup, then set the **plan.budget.write_hold_budgets** app option to true.
- Write budget SDC with **write_budgets**
- Overlay the output from **write_budgets** into your block to set boundary constraints for the block.

It is only necessary to apply your internal block constraints one time. After applying the constraints, you can incrementally apply new boundary constraints whenever your budget is updated. If you do not have internal block constraints, consider using the **split_constraints** command to derive block-level constraints based on chip-level constraints.

You can make **write_budgets** create "full" budgets, which include constraints for both the boundary of the block and internal constraints. To write full budgets, specify the **-full_budget_blocks** option. In general, it is not recommended to use full budget constraints, because there might be a large runtime and memory impact. Also, applying full budgets is disruptive because it overwrites any existing constraints for your block. Incremental budgets allow the existing internal constraints of your block to be maintained. See the section on "Budget Constraint Styles" for more information.

The budget constraints written by **write_budgets** include clock latency adjustments to account for partially implemented clock networks. Different latency adjustments should be chosen at different stages of the design flow. See the details in the description of the **-adjust_latency** option of the **set_budget_options** command. Also, you should read the description of the **-latency_target** and **-balance** options of the **compute_budget_constraints** command.

Clock Name Mapping

It is possible that clocks defined in your blocks have names that do not match connected top-level clock names. For example, a top-level clock named "SYS_CLK" may be connected to a block-level clock named "CLK". The **write_budgets** command understands that names may not match, and automatically derives the mapping between top-level and block-level clocks. This allows **write_budgets** to generate the proper constraints for the block.

If there is insufficient information to automatically derive a mapping between top-level clocks and block-level clocks, you can manually specify a mapping with the **set_block_to_top_map** command. If no mapping is specified, **write_budgets** assumes that the block-level clock name matches the top-level clock name.

Budget shells

The **write_budgets** command supports a construct called "budget shell", which is a special library cell without a netlist. In a design hierarchy, a budget shell can be used to replace a lower-level block. After the lower level block is replaced by a shell, you can no longer examine timing for paths inside of the block or paths that pass between the block and its parent block. Instead, the boundary ports of a "budget shell" are budgeted: paths that originally went from the parent hierarchy into child hierarchy are replaced with new budgeted paths that go from parent hierarchy to the ports of the shell. The timing of the new paths is adjusted to match the budget constraints that you set for the replaced block.

Budget shells are useful for separating the higher levels of your design from the lower levels of your design. If you use a budget shell for a lower level design, you can optimize the higher level design independently. The timing of the top-level design only depends on the lower level's budget constraints, not on the actual timing in the lower level.

Note that budget shells do not eliminate the need to optimize the top level along with the full (non-shell) block. Because the shell only follows a budget, it is inherently pessimistic. Using the actual timing of a lower-level block might let you see additional slack along a path that the budget did not see. In particular, it is not advisable to use budget shells when doing hold fixing on paths between top level and block level. The budget pessimism potentially causes over-buffering.

Hierarchical budgeting

The **write_budgets** command has advanced options to control how the boundaries between a logical or physical hierarchy are handled for timing. This section first describes the default behavior of **write_budgets** and then discusses the variants of that behavior. For the purpose of discussion, assume a design with three block levels: top, mid, and bot. Everything except the top must be declared as budget blocks by using:

```
prompt> set_budget_options -add_blocks {mid mid/bot}
```

By default, **write_budgets** works on all block hierarchy instances declared with **set_budget_options**. For each block in the hierarchy, constraints are written to characterize the timing paths and clocks that enter at the block's inputs or leave at the block's outputs. For example, block mid will get boundary constraints for inputs and outputs coming down from or going up to block top. And block bot will get boundary constraints for inputs and outputs coming down from or going up to block mid. No boundary constraints will be written for block top.

You can specify "budget shell" blocks with the **-shell_subblocks** option of **write_budgets**. See the section on "budget shells" for more information. Because no logic in a budget shell hierarchy is exposed to its parent block, different constraints are needed for that parent block. For example, if "bot" was listed as a shell block, the constraints for "mid" would not include full paths that pass between mid and bot. Instead, constraints are added that constrain the partial paths from mid to the ports of bot. The timing constraint will be created based on your current budget constraints, not the current timing of the full path. Also, the budgeted paths between "top" and "mid" will also be constrained -- as they were in the default behavior of **write_budgets**. In summary, "mid" sees both budgeted path constraints passing from/to "top" and budgeted path constraints passing from/to "bot".

You can specify that nested blocks in a design hierarchy will not be abstracted when optimized in their parent context with the **-design_subblocks** option of **write_budget**. This allows a child block and its parent block to be optimized together in a "virtual flat" manner. Because all of the logic of the child block is exposed to its parent block, different constraints are needed for that parent block. For example, if "bot" was listed as a flat block, the constraints for "mid" would need to cover not only all of the paths of mid, but also all the paths of bot.

In a multilevel hierarchy, the **-design_subblocks** options can force constraints to cover many levels at the same time. For example, if you are generating constraints for "top", and block "mid" is labeled as a design_subblock, then the constraints for top will need to consider boundary constraints for "bot". In the following example, the generated constraints for the top level include all paths at the top, all paths in mid, and the boundary paths of bot.

```
prompt> write_budgets -top design_subblocks mid
```

Budget Constraint Styles

For each budgeted block in your design, the **write_budgets** command can generate budgets in one of two styles. Budgets can include "full constraints" or budgets can include "incremental constraints". Full constraints are a complete set of constraints for a block. Incremental constraints preserve a block's internal constraints but updates the constraints on the block interface. The budget style is controlled by the **-full_budget_blocks** option.

"Incremental" constraints apply only to the interface timing paths of a budgeted block. These constraints are intended to be overlaid on top of the existing internal constraints of the block. Using these constraints, you can incrementally change the block's interface boundary constraints without disturbing the internal constraints. If you do not have existing internal constraints for your block, you can generate them by using the **split_constraints** command. These constraints are generated by default.

"Full" constraints include constraints for both the interface of the budgeted block *and* the internal block paths. If you use these constraints, any existing constraints on the block will be removed and replaced. These constraints are used for the top-level block budget, for modules selected by the **-full_budget_blocks** option and for parent blocks of modules listed in the **-shell_subblocks** option.

When calculating "full" constraints, the **write_budgets** command examines the timing of all internal paths for each budgeted block. If you are using abstracts to represent some of your blocks, timing the internal paths of the block is not possible. In this case, "full" constraints cannot be generated for abstracted blocks. If you want to generate "full" constraints, you must also use a full design view (not an abstract) for the block.

Memory Considerations

When you generate full constraints, you are required to change the selected blocks to a full design views. For large designs, this could require significant memory and runtime. To reduce the resource requirements, consider using the **split_constraints** command to generate the internal constraints of your budgeted block. The **split_constraints** command is specifically designed to use much less memory and runtime. After you have the results of **split_constraints**, you run **write_budgets** its default incremental setting.

This command returns 1 on success, 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example creates block budget scripts for all declared budget blocks. The output is written to a directory called "budgets". If the directory already exists, it is overwritten.

```
prompt> write_budgets -force
```

The following examples creates budget scripts for the instance i_cpu in all modes. The output is written to a directory called "cpu_budget".

```
prompt> write_budgets -block i_cpu -output cpu_budget
```

The following example assumes a pre-CTS design. The commands automatically create a budget that assumes clock latency targets should be balanced across all blocks. Make sure you are using "prects" latency adjustment for your blocks. Then write the budget budgets along with the necessary clock latency adjustments to implement the balancing.

```
prompt> compute_budget_constraints -latency_targets estimated -balance true
prompt> set_budget_options -adjust_latency prects -all
prompt> write_budgets
```

The following example assumes a design where CTS has been run in the blocks but not yet balanced at the top. The commands automatically create a budget that assumes clock latency targets should be balanced across all blocks. Make sure you are using "target" latency adjustment for your blocks. Then write the budget budgets along with the necessary clock latency adjustments to implement the balancing.

```
prompt> compute_budget_constraints -latency_targets actual -balance true
prompt> set_budget_options -adjust_latency target -all
prompt> write_budgets
```

The following example assumes a design that has the final block and top clock trees. The commands create a budget that uses the actual current latencies (with no rebalancing). Make sure you are using "actual" latency adjustment for your blocks. Then write the budget budgets along with the necessary clock latency adjustments at block inputs.

```
prompt> compute_budget_constraints -latency_targets actual -balance false
prompt> set_budget_options -adjust_latency actual -all
prompt> write_budgets
```

SEE ALSO

compute_budget_constraints(2)
report_block_to_top_map(2)
report_budget(2)
set_block_to_top_map(2)
set_pin_budget_constraints(2)
set_boundary_budget_constraints(2)
set_latency_budget_constraints(2)
set_budget_options(2)
set_budget_margins(2)
set_budget_shell_latencies(2)
split_constraints(2)
write_script(2)
plan.budget.all_design_subblocks(3)
plan.budget.all_full_budgets(3)
plan.budget.pessimistic_driving_cells(3)
plan.budget.write_hold_budgets(3)

write_busplans

Writes various files and scripts for busplans.

SYNTAX

int **write_busplans**
[-implementation_script *filename*]
[-multi_cycle_script *filename*]
[-xml_file *filename*]
[buses]

Data Types

filename string
buses collection of busnames

ARGUMENTS

-implementation_script filename

Specifies the name for the implementation script. This script contains constraints, such as movebounds and route corridors, that are used to create the busplan in the design.

-multi_cycle_script filename

Specifies the name for the multi cycle path script. This script sets multi cycle path constraint on paths that have virtual registers.

-xml_file filename

Specifies the name for an XML data file. This saves all current bus planning information (for the specified busplans) into an XML document. These XML documents can be loaded into the tool with the *load_busplans* command.

buses

Specifies the names of the buses to consider. If no buses are specified, the default is all buses in the busplan.

DESCRIPTION

Creates various files and scripts regarding the busplan. Use options to create the bus plan representation in different formats.

SEE ALSO

create_busplans(2)
get_busplans(2)
get_net_estimation_rules(2)
load_busplans(2)
modify_busplan(2)
report_busplans(2)
set_net_estimation_rule(2)

write_cell_expansion

Writes the cell expansion data to a file to support ASCII flow.

SYNTAX

int **write_cell_expansion**
[-output *output_file_name*]
[-scale_factor *unit_scale*]

Data Types

output_file_name string
unit_scale float

ARGUMENTS

-output *output_file_name*

Specifies the name of the output expansion data file. This is a required argument.

-scale_factor *unit_scale*

Specifies the scale factor between tools. This is optional.

DESCRIPTION

The **write_cell_expansion** command writes the cell expansion data to a file.

EXAMPLES

The following example writes out a cell expansion file from the existing design.

prompt> **write_cell_expansion -output out.exp**

SEE ALSO

read_cell_expansion(2)

write_checksum

This utility is for generating checksums.

SYNTAX

status **write_checksum**
[-output_directory *output_directory_path*]
[-type *checksum_type_list*]
[-scenario *scenario_name*]

Data Types

output_directory_path string
checksum_type_list list
scenario_name list

ARGUMENTS

-output_directory *output_directory_path*

Directory in which checksum will be generated.

If this option is not specified, current directory is utilized.

-type *checksum_type_list*

Specifies the list of checksums to be generated.

Valid values it can take are **all**, **library**, **design**, **parasitics**, and **constraints**.

If this option is not specified, all checksums are generated.

-scenario *scenario_name*

Specifies the scenario for which checksum needs to be generated.

If this option is not specified, the tool uses the current scenario.

DESCRIPTION

The **write_checksum** command generates checksums for input consistency checker.

The **write_checksum** command generates the checksums. If the **-output_directory** option is not specified, the checksums are

generated in the current directory. The checksum is generated into a subdirectory with the name of the scenario inside output_directory. If scenario option is not specified then the current_scenario is utilized.

If the **-type** option is not specified, all the checksums are generated. The values it can take are **all**, **library**, **design**, **parasitics**, and **constraints**.

EXAMPLES

The following example generates checksums on the current design to the checksum_out directory.

```
prompt> write_checksum -output_directory checksum_out
```

SEE ALSO

compare_checksum(2)

write_clock_trunks

Writes a Tcl script to re-create the specified clock trunk or for all clock trunks. The script includes clock trunk guide buffers.

SYNTAX

write_clock_trunks
[-clock *clock_list*]
[-file *file_name*]

Data Types

clock_list list or collection
file_name string

ARGUMENTS

-clock *clock_list*

Writes the clock trunk only for the specified clocks. By default, clock trunks for all clocks in the design are written.

Use this option to save a planned clock trunk for a specific clock, then re-create the clock trunk in a newly modified netlist to handle ECOs.

-file *file_name*

Specifies the name of the Tcl file to write. By default, the Tcl file is written to the terminal and to the logfile.

DESCRIPTION

This command writes a Tcl script you can use to re-create the current clock trunk.

This command does the following:

- The clock trunk for the given clocks is determined following the rules as for the **gui_load_clock_trunk_planning** command.
- For any buffers in the clock trunk, Tcl code is generated which re-creates the buffer if not present, and sets the location and placement status to its current value.
- For other cells in the clock trunk, Tcl code is generated which sets the location and placement status of the cell to its current value, if the cell exists.

EXAMPLES

The following example writes a script that can be used to re-create the clock trunk for the clock named CLK.

prompt> **write_clock_trunks -clock CLK -file CLK_trunk.tcl**

SEE ALSO

gui_load_clock_trunk_planning(2)
remove_clock_trunk_endpoints(2)
report_clock_trunk_endpoints(2)
set_clock_trunk_endpoints(2)

write_collection

Output a machine readable report of attribute values for elements in a collection.

SYNTAX

write_collection
collection
-file *filename*]
[-format *format*]
[-max_rows *value_count*]
[-columns *attribute_list*]
[-metadata]

string *collection*
string *filename*
string *format*
int *value_count*
list *attribute_list*

ARGUMENTS

collection

Specifies the collection on which to report. The collection may be homogeneous or heterogeneous. If the collection is very large, you may want to use the **-max_rows** option to limit the size of your output.

-file *filename*

This option indicates the name of the file to which the data is written.

-format *format*

This option accepts one of the following valid argument values: "csv" | "tsv". If the option is omitted, the default format is "csv", or command separated values. The argument "tsv" produces a tab separated values formatted data.

-max_rows *value_count*

This option indicates how many rows of data to include in the output. This number indicates the number of collection elements on which data is output.

-columns *attribute_list*

Indicates the set of attributes for which to output values, and their order in the output.

The special attribute name "object_class" may be used to include the element object class names, such as "cell" or "net", in the output.

If the option is omitted, then the object names and object class (or type) names are output by default.

-metadata

This option indicates that a meta data section should be included in the output.

DESCRIPTION

This command outputs tabular data of attribute values for elements of the given collection. The attribute values in the output are determined by the list of attributes given to the **-columns** option. The columns in the tabular report will be ordered by the order of attributes in the list. The command **list_attributes** may be called to view a list of attributes defined for an object class.

If the collection is large, the output size may be limited by indicating a **-max_rows** value. The commands **filter_collection** and **sort_collection** may be called to filter and sort a collection based on some criteria prior to creating a report for the collection elements.

By default, the output file omits the meta data section. If **-metadata** option is present, the output initiates with the meta data section which contains information such as the file generation timestamp, the product name and version and column data types. Some consumers of csv/tsv files benefit from the extra information in the meta data while other consumers do not handle it well.

EXAMPLES

The following example from IC Compiler II writes the full_name and area values of cells in a collection to a comma separated values (CSV) file named "cell_area.db".

```
icc2_shell> set cells [get_cells U*]
icc2_shell> write_collection $cells -columns {full_name area} \
        -file cell_area_pins.db
```

The next example outputs the same values to a tab separated values (TSV) file.

```
icc2_shell> write_collection $cells -columns {full_name area} \
        -format "tsv" -file cell_area_pins.db
```

The next examples limits the output to a CSV file to the first 10 objects in the collection.

```
icc2_shell> write_collection $cells -columns {full_name area} \
        -file cell_area_pins.db -max_rows 10
```

The next example first sorts the original collection of cells by the area attribute value in descending order, limiting the resulting collection to the top 10 area values. The example then outputs the full_name and area values to a CSV file named top_10_areas.db.

```
icc2_shell set sorted_cells [sort_collection -dictionary \
        $cells {area- full_name} -limit 10]
icc2_shell> write_collection $cells -columns {full_name area} \
        -file top_10_areas.db
```

The next example writes a sorted collection as in the above example but further limits the output to the first 10 objects in the collection. Note that this command may produce an output that is different from the above example. Limiting the sort to the first 10 values may produce a collection with more than 10 objects since multiple objects may share the same area value.

```
icc2_shell set sorted_cells [sort_collection -dictionary \
        $cells {area- full_name} -limit 10]
icc2_shell> write_collection $sorted_cells -columns {full_name area} \
        -file first_10_objects_by_area.db -max_rows 10
```

The next example outputs to a CSV file which will include the meta data section.

```
icc2_shell> write_collection $cells -columns {full_name area} \
         -format "csv" -file cell_area_pins.db -metadata
```

SEE ALSO

collections(2)
list_attributes(2)
filter_collection(2)
sort_collection(2)
report_collection(2)

write_def

Writes a DEF file from a given design.

SYNTAX

string **write_def**
[-compress *method*]
[-units *units_per_micron*]
[-include *include_list*]
[-exclude *exclude_list*]
[-objects *objects*]
[-version *version*]
[-convert_sites { {*from_site to_site*} ... }
[-include_tech_via_definitions]
[-via_as_fixed]
[-routed_nets]
[-self_contained]
[-bus_delimiters *bus_bit_chars*]
[-traverse_physical_hierarchy]
[-no_marker_layer]
def_file_name
[-include_physical_status *include_physical_status_list*]
[-exclude_physical_status *exclude_physical_status_list*]
[-only_master_variant]
[-cell_types *cell_types_list*]
[-net_types *net_types_list*]
[-include_pad_owned_terminals]
[-design *design*]

Data Types

design	string
method	string
units_per_micron	int
include_list	list
exclude_list	list
objects	collection
version	string
from_site	string
to_site	string
bus_bit_chars	string
def_file_name	string
include_physical_status_list	list
exclude_physical_status_list	list
cell_types_list	list
net_types_list	list

ARGUMENTS

-compress ***method***

Specifies file compression format. The only format currently supported is **gzip**. A .gz extension is appended to the file name if no extension is specified. By default, DEF files are written in uncompressed text format.

-units ***units_per_micron***

Specifies the number of units per micron in the DEF file. All database length units are scaled accordingly in the output DEF file. If not specified, the units per micron is obtained from the technology length precision. Valid values are 100, 200, 1000, 2000, 10000, and 20000.

-include ***include_list***

Specifies which optional constructs and object types to include in the output DEF file. The output DEF file always contains the basic DEF items: version, divider character, bus bit characters, design name, units, and die area. When you use the **-include** option, only the listed optional constructs and object types are included in the generated DEF file. These are the optional constructs and object types:

blockages
bounds
cells
fills
nets
pg_metal_fills
ports
routing_rules
rows_tracks
scanchains
specialnets
vias
cut_metal

-exclude ***exclude_list***

Specifies which optional constructs and object types to exclude from the output DEF file. When you use this option, all of the optional constructs and object types are included in the generated DEF file except for those specified in the exclusion list.

When you use neither the **-include** nor **-exclude** option, the default behavior is to write out all the optional constructs and object types.

-objects ***objects***

Specifies a collection of objects to write out to the DEF file. The objects can be physical cells, physical nets, physical ports, routing_blockage (non-reserved), placement_blockage (hard, soft, partial), move bounds, rows, net shapes and vias. The output DEF file contains only the objects specified by this option, unless you also use the **-include** or **-exclude** option.

- If cells are specified, they are written to the COMPONENTS section, and all other cells are omitted.
- If nets are specified, they are written to the SPECIALNETS and/or NETS sections, and all other nets are omitted.
- If ports are specified, they are written to the PINS section, and all other ports are omitted.
- If blockages are specified, they are written to the BLOCKAGES section, and all other blockages are omitted.
- If move bounds are specified, they are written to the REGIONS and GROUPS sections, and all other move bounds are

omitted.

- If net shapes and/or vias are specified, they are written to the SPECIALNETS and/or NETS sections, and all other shapes and/or vias and the net connections are omitted. You can use the **set_app_options -name file.def.skip_connection_of_incomplete_net -value false** command before write_def command to write the net connections.
- If you specify the lower left via of a via ladder to the **-objects** option, the via ladder is written out. If you specify other vias or shapes of a via ladder to the -objects option, the via ladder is excluded from the generated DEF file, and only the corresponding net will be written out.

-version *version*

Specifies the DEF syntax version. Valid values are 5.7 and 5.8. The default is DEF version 5.8.

-convert_sites { {*from_site to_site*} ... }

Specifies a list of site name pairs, with each pair representing a map from one site name to another site name. All rows of site *from_site* are written as rows of *to_site*.

-include_tech_via_definitions

Includes all via definitions in the technology file, including those not used in the design, in the output DEF file. By default, only via definitions present in the design are written out to the DEF VIAS section.

-self_contained

Indicates that the output DEF must be self-contained. A self-contained DEF should not reference any objects which are not defined in the same DEF file. Please note that this option supports cell and move bound objects only.

-via_as_fixed

Indicates that all the via definitions are written out as fixed vias in the output DEF file.

-routed_nets

Indicates that all the nets written out in the output DEF file are routed nets i.e. has atleast one shape associated it.

-bus_delimiters *bus_bit_chars*

Specifies the bus bit characters to use in the output DEF. Valid values are "[]", "{}", "()", and "<>". When specified, the DEF BUSBITCHARS statement and bus names are written out using these bus bit characters.

-traverse_physical_hierarchy

Indicates writing the design information across all levels of the physical hierarchy to DEF file.

nets
: If the net crosses physical hierarchy, the sub-block nets are not written out, but all their wiring shapes are superimposed onto the net and written out. All the nets, which does not cross physical hierarchy, can be written out.

pins
: Sub-block pins are not written out, but all their shapes can be superimposed onto the connected net and written out.

via definitions and bounds
: If there is a name collision between top and sub block objects, the sub block objects are written out with block name prefix.

routing rules If a routing rule exists in the sub block and matches a routing rule that already exists in the top block, the sub block routing rule is not written out. If two routing rules have the same name but they are different in fact, the sub block routing rule is written out with the block name prefix.

cells, rows, blockages and tracks
They can be written out from each level of the physical hierarchy.

-no_marker_layer

Indicates disabling the writing of marker layers.

def_file_name

Specifies the name of the DEF file to be written. If the file already exists, it is overwritten. If the name ends with the .gz extension, it is compressed using the gzip format.

-include_physical_status *include_physical_status_list*

Specifies which cells to include in the output DEF file based on their physical status. When you use this option, all of the specified cells are excluded in the generated DEF file except for those whose physical status is specified in the *include_physical_status_list*. These are the optional physical status:

unplaced
placed
legalize_only
application_fixed
fixed
locked
all

-exclude_physical_status *exclude_physical_status_list*

Specifies which cells to exclude in the output DEF file based on their physical status. When you use this option, all of the specified cell are included in the generated DEF file except for those whose physical status is specified in the *exclude_physical_status_list*.

-include_physical_status and *-exclude_physical_status* are mutually exclusive.

-only_master_variant

Indicates that all the instances of variants are replaced by instances of the masters.

-cell_types *cell_types_list*

Specifies which cells to include in the output DEF file based on their types. When you use this option, all of the specified cells are excluded in the generated DEF file except for those whose cell types is specified in the *cell_types_list*. These are the specifiable cell types:

corner
end_cap
pad
flip_chip_pad
flip_chip_driver
pad_spacer
macro
lib_cell

This option provides additional level of filtering when used along with *-include_physical_status* and *-exclude_physical_status*.

-net_types *net_types_list*

Specifies which nets to include in the output DEF file based on their types. When you use this option, all of the specified nets are excluded in the generated DEF file except for those whose net types is specified in the *net_types_list*. These are the specifiable net types:

signal
power
ground
clock

-include_pad_owned_terminals

Includes port terminals having **is_owned_by_pad** attribute values of true. If the port **preferred_pin** attribute is set, then the preferred pin terminal is output. Otherwise if the port **die_side** attribute is set, then terminals on the die side are output. Otherwise all such terminals are output. Without this option, such terminals are output if and only if their **is_def_duplicate** attribute values are true.

-design *design*

Specifies the design (block) to be written as a DEF format. The default is the current block.

DESCRIPTION

This command writes the current or given design to a DEF file, including the physical layout, netlist, and design constraints. If the current design is not linked, it is automatically linked and then written out.

For the shapes which have no nets association, only when they are cut metal, or special wire extension, or metal fill they will be written out, otherwise, they will not.

The **-include** and **-exclude** options are mutually exclusive. When you use the **-include** option, the DEF contains the basic DEF items (version, divider character, bus bit characters, design name, units, and die area) and the objects specified with this option. When you use the **-exclude** option, the DEF contains everything except for the objects specified with this option.

When neither the **-include** nor **-exclude** option is used, the default behavior is to write out everything. However, if you use the **-objects** option, the output includes only the objects specified in the **-objects** argument.

These are the valid values for the *include_list* and *exclude_list*:

blockages Routing and placement blockage definitions defined in the DEF BLOCKAGES section.

bounds Move bound definitions defined in the DEF REGIONS and GROUPS sections.

cells Cell definitions defined in the DEF COMPONENTS section.

fills Floating metal fills defined in the DEF FILLS section.

nets Regular net definitions defined in the DEF NETS section.

pg_metal_fills Special net power and ground net fill shapes defined in the DEF SPECIALNETS section.

ports Port definitions defined in the DEF PINS section.

routing_rules Nondefault routing rule definitions in the DEF NONDEFAULTRULES section.

rows_tracks Row and track definitions defined with the DEF ROW and TRACKS statements.

scanchains Scan chain definitions in the DEF SCANCHAINS section.

specialnets Special net definitions, except power and ground net fill shapes, in the DEF SPECIALNETS section. Power and ground net fill shapes are specified by the option value, pg_metal_fills.

vias Via definitions defined in the DEF VIAS section.

cut_metal Shapes on the cut-metal layers, and metal shapes under cut-metal shapes.

These are the valid value only for the *exclude_list*:

non_mask_purpose Shapes of LayerDataType with nonMask=1 will be excluded from the DEF.

EXAMPLES

The following example writes a compressed DEF file using 1000 units per micron:

prompt> **write_def -compress gzip -units 1000 top.def**

The following example writes a DEF file that includes the floorplan data and excludes all other optional constructs and object types.

prompt> **write_def -include {rows_tracks bounds cells blockages} fp.def**

The following example writes a DEF file that excludes all special wiring but includes all other optional constructs and object types.

prompt> **write_def -exclude {specialnets pg_metal_fills} no_special.def**

SEE ALSO

read_def(2)

set_app_options(2)

write_default_pg_pattern

Writes out the default **create_pg_*_pattern** commands to describe the default PG patterns for the current design. Supported patterns are PG ring, mesh, macro connection, standard cell connection, and special patterns.

SYNTAX

status **write_default_pg_pattern**
[-type {*pattern_types*}]
[-output_filename *file_name*]

Data Types

pattern_types list
file_name string

ARGUMENTS

-type {*pattern_types*}

Specifies a list of one or more PG pattern types to write out. Supported types are: **ring**, **mesh**, **macro_conn**, **std_conn**, and **special**. By default, the command outputs all PG pattern types.

-output_filename *file_name*

Specifies the file name to use to write out the patterns. By default, the command writes the output to the terminal.

DESCRIPTION

This command writes out the appropriate **create_pg_*_pattern** commands to describe the default PG ring, mesh, macro connection, standard cell connection and special patterns. The command writes the output to the terminal and saves the output to the file specified by the **-output_filename** option.

EXAMPLES

The following example writes out the default mesh and ring patterns to *default.tcl* and to the terminal.

```
prompt> write_default_pg_pattern -type {mesh ring} \
            -output_filename default.tcl
```

SEE ALSO

create_pg_macro_conn_pattern(2)
create_pg_mesh_pattern(2)
create_pg_ring_pattern(2)
create_pg_special_pattern(2)
create_pg_std_cell_conn_pattern(2)

write_design_io

Writes out a text file containing information for I/O elements present in the design.

SYNTAX

string **write_design_io**
-file_name *file_name*
[-contents *object_type*]
[-columns_file *file_name*]
[-map_file *file_name*]
[-delimiter_char *char*]
[-cell_origin_type *string*]
[-within *bbox*]
[-objects *collection*]
[-units *units_per_micron*]

Data Types

file_name string
object_type string
char string of one character
bbox list of coordinates
units_per_micron integer

ARGUMENTS

-file_name *file_name*

Specifies the name of the output file to write the I/O design data. The file is an ASCII file that uses comma-separated value (CSV) format. A different field separator can be specified with the **-delimiter_char** option.

-contents *object_type*

Specifies the I/O object types to write to the output file. Supported object types are "c4", "ubump", "tsv", and "all". If not specified, the command writes out all I/O object types.

-columns_file *file_name*

Specifies a file that contains the column names to write out. Use this file to specify the column order and limit the data that is written to the output file.

For example, the following columns file specifies that only X origin, Y origin, C4 instance name, TSV instance name, and reference name information are written to the output file.

```
X_origin,Y_origin,C4_inst,TSV_inst,Reference_name
```

-map_file *file_name*

Specifies a CSV file that maps custom column names in the output CSV file to standard Synopsys column names. Custom column names are specified in the first field, and standard Synopsys column names are specified in the second field, separated by a comma.

For example, the following map file maps four custom column names to their corresponding Synopsys column names.

```
IO_driver name,C4_inst
Ref_Cell,Reference_name
X_COORD,X_origin
Y_COORD,Y_origin
```

-delimiter_char *char*

Specifies a single character to be used as the field delimiter in the output files. If not specified, comma is the field delimiter.

-cell_origin_type *string*

Specifies what point of the cell should be considered as its origin in order to do its placement. The allowed values are *lower_left* or *center*. The bounding box of the cell will be used to calculate its center. This option is optional. If not specified, the lower left corner of the cell will be considered.

-within *bbox*

When this option is specified, the writer will select only those I/O objects which are contained within the given bbox and write them out. This option can also be specified along with the *-contents* option.

-objects *collection*

When this option is specified, the writer will write out only those I/O objects which are a part of the collection. If the collection contains objects which are not C4 bumps, ubumps or TSVs, then such objects will be skipped, and the writer will continue to write the remaining objects. This option cannot be specified with *-within* option or with *-contents* option.

-units *units_per_micron*

If the length unit used in the output CSV file needs to be different from that of the design library, this option can be used to specify the custom unit for length. When specified, the distance values (of X/Y coordinates) in microns will be multiplied by this number and written out in the CSV file. The value must be an integer >=1. This is an optional option. If not specified, the distance values will be assumed to be in the same unit as the current library.

DESCRIPTION

The **write_design_io** command writes out the I/O information for to a CSV file. The file can be read directly by the **read_design_io** command. The command will write the following data into the output file:

- C4 bump cells

- Microbump cells

- Through-Silicon Vias (TSVs)

Each row in the output file contains information for one I/O object. By default, each row contains the name of the object, its reference name, X/Y location, orientation, net, and port name if it is connected. By default, the following standard columns are written to the output file.

- Reference_name

- C4_inst
- Ubump_inst
- TSV_inst
- X_origin
- Y_origin
- Orientation
- Port_name
- Net_name
- Comments

The columns and column order written to the file can be specified in a file and referenced by the **-columns_file** option. Custom column names can be specified by creating a mapping file and referencing the file with the **-map_file** option. The default field delimiter is a comma and can be changed by specifying the **-delimiter_char** option.

Field Map File Details

If the output file should contain custom names for fields instead of the Synopsys standard names, a mapping file can be provided via the **-map_file** option. Use the mapping file to create a map between your custom name and the corresponding Synopsys standard name for the field.

Each line in the map file contains exactly two strings separated by a comma. The first string is a user-defined field string and the second string is a Synopsys keyword. The order of lines does not matter. If the second string is not recognized, the line is ignored with a warning message.

The following map file maps the standard C4_inst field name to "IO_driver", and maps the standard Reference_name field name to "Ref_Cell".

```
IO_driver,C4_inst
Ref_Cell,Reference_name
```

The custom field names should not be same as the standard field names. Also, they must not contain any wildcard or special characters.

EXAMPLES

The following example writes all I/O design objects at the top level to a CSV file named out1.csv. All standard fields all written to the output file.

```
prompt> write_design_io -file_name out1.csv
1
```

The following example writes out a colon-separated file with custom column names, specifies a map file for the column names, specifies what columns to write out. Only TSVs are written out.

```
prompt> write_design_io -delimiter_char ":" \
  -map_file map.txt -contents "tsv" -columns_file cols.txt \
  -file_name out2.csv
Warning: Ignoring entry as number of columns in map file should be exactly 2.
 (map.txt line 3) (TLNL-014)
```

```
Warning: Unrecognized column name 'DUMMY' in map file, ignoring it.
  (map.txt line 4) (TLNL-018)
Parsing columns file cols.txt
1
```

SEE ALSO

create_3d_mirror_bumps(2)
create_bond_pad_array(2)
create_bump_array(2)
create_tsv_array(2)
derive_3d_interface(2)
read_aif(2)
read_design_io(2)

write_dff_trace_filters

Writes the current trace filter pattern set to a file.

SYNTAX

status **write_dff_trace_filters**
-type pin | net
-filename *filename*
[-overwrite]

Data Types

filename string

ARGUMENTS

-type pin | **net**

Specifies the type of filter set to save to a file, either pin or net.

-filename ***filename***

Specifies the filename to use when saving the indicated filter set type.

-overwrite

Overwrites the file if it already exists.

DESCRIPTION

This command writes out trace filter patterns created by the **compute_dff_connections** command. Trace filter patterns are written to the filename specified by the **-filename** option. This file can be read by the **create_dff_trace_filters** command to restore the list of filters in another session.

EXAMPLES

The following example saves pin filters to a file called pinFilters.txt, and reads the filters with the **create_dff_trace_filters** command.

```
prompt> write_dff_trace_filters -type pin -filename pinFilters.txt
1
prompt> create_dff_trace_filters -type pin -filename pinFilters.txt
1
```

SEE ALSO

compute_dff_connections(2)
create_dff_trace_filters(2)
remove_dff_trace_filters(2)

write_drc_error_data

Writes a DRC error data object associated with a block to file on disk and returns a collection that contains the DRC error data object.

SYNTAX

collection **write_drc_error_data**
  -error_data *drc_error_data*
  -file_name *file_name*
  [-overwrite]

Data Types

drc_error_data collection
file_name string

ARGUMENTS

-error_data *drc_error_data*

Specifies the DRC error data object to export to a file.

-file_name *file_name*

Specifies the name of the error data file to create.

The naming convention for an error data file is to append an underscore, the checker name, and the ".err" suffix to the design name. For example, for a design named "my_design" and a checker named "test", the error data file name is named "my_design_test.err".

-overwrite

Overwrites an existing file with the same name as the specified output file.

By default, the command fails is a file exists with the same name.

DESCRIPTION

The **write_drc_error_data** command writes an error data object to an external error data file with the specified file name. The command fails if the specified error data file already exists.

The original error data object remains open and in memory. The external error data file is not opened.

The command returns 1 if successful; 0 otherwise.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example exports the physical DRC error data file that is named "dppinassgn.err". Although the output looks like a list, it is not. The output is a display of the name of the object in the returned collection.

```
prompt> write_drc_error_data -file_name my_design_dppinassgn.err -overwrite
1
```

SEE ALSO

create_drc_error_data(2)
open_drc_error_data(2)
close_drc_error_data(2)
save_drc_error_data(2)
get_drc_error_data(2)
remove_drc_error_data(2)
collections(2)

write_ems_rules

Saves commands used to create user-defined rules as a script.

SYNTAX

write_ems_rules
[-rule *ems_rule_name*]
[-all]
[-overwrite]
output_file

Data Types

ems_rule_name string
output_file string

ARGUMENTS

-rule *ems_rule_name*

Specifies the rule name for an EMS rule. The option takes name of a rule or a pattern capturing names of multiple rules as argument. The *ems_rule_name* should be already available in EMS memory, otherwise the command will fail with an error message. The **-rule** option is optional, and the **-all** and **-rule** options are mutually exclusive.

output_file

Specifies the output file name. The command writes out a set of Tcl commands to create one or more EMS rules to the specified file, and the file can be used as a Tcl script to source commands to create user-defined EMS rules in a new session. If the file name already exists, the tool issues an error message. To overwrite the contents of an existing file, use the **-overwrite** option. The *output_file* argument is a mandatory argument.

-all

This option saves the commands to create all user-defined rules in EMS memory into the user-provided file as a Tcl script. If command is not provided with either **-all** or **-rule** options then the default behavior is as if **-all** option is provided. The option **-all** is an optional option and is mutually exclusive with the **-rule** option.

-overwrite

Forces overwrite of the output file if the file already exists. By default, if the file exists, an error message is issued and the command exits. This is an option is optional.

DESCRIPTION

This command is used to write out a Tcl script file that contains commands to create user-defined rules. This script can be sourced in a different session to create the user-defined rules again; this is needed since user-defined rules are not persistent across sessions.

EXAMPLES

The following example creates, edits, writes out, and reads in EMS rules in a two sessions.

```
# In session 1
prompt> create_ems_rule -name "TEMP-001" \
  -severity "Info" -message "Pin:%pin has capacitance:%cap"
prompt> edit_ems_rule -name "TEMP-001" \
  -parameters "name:pin type:string command:get_pins"
prompt> edit_ems_rule -name "TEMP-001" \
  -parameters "name:cap type:double"
prompt> create_ems_rule -name "TMP-002" \
  -severity "Error" -message "Cell %cell is not placed"
prompt> edit_ems_rule -name "TMP-002" -severity "Info"
prompt> edit_ems_rule -name "TMP-002" \
  -message "The Cell %cell is not placed."
prompt> edit_ems_rule -name "TMP-002" \
  -parameters "name:cell type:string command:get_cells"
prompt> write_ems_rules -overwrite -all ems_rules.tcl
prompt> quit

# In session 2
prompt> source ems_rules.tcl
prompt> report_ems_rules -all

Rule: TEMP-001
  Severity: Info
  Message: Pin:%pin has capacitance:%cap
  Parameter name: cap type: double command: NA
  Parameter name: pin type: string command: get_pins

Rule: TMP-002
  Severity: Info
  Message: The Cell %cell is not placed.
  Parameter name: cell type: string command: get_cells
```

SEE ALSO

create_ems_rule(2)
report_ems_rules(2)

write_feasibility_constraints

Writes timing constraints for the current design to make paths feasible by using a slack threshold value.

SYNTAX

status **write_feasibility_constraints**
-output *dir_name*
[-feasibility_factor *feasibility_factor_value*]
[-method multicycle_path | false_path | path_margin]
[-type setup | hold | both]
[-effort medium | high]
[-apply]
[-compress none | gzip]

Data Types

dir_name string
feasibility_factor_value float

ARGUMENTS

-output *dir_name*

Specifies the name of the directory **dir_name** in which to write the timing constraint files for the current design.

-feasibility_factor *feasibility_factor_value*

Specifies the value used to determine the feasible paths. By default, the value is set to 2.0. A timing path having negative slack, at the data-pin of an endpoint is said to be an infeasible timing path. This slack threshold value is determined by the user such that the absolute slack value for an endpoint should be lesser than the product of **feasibility_factor_value** and the capture clock period. If the endpoint belongs to an IO bound path, then the threshold is adjusted by subtracting the product of IO constraint value and the feasibility factor.

-method multicycle_path | **false_path** | **path_margin**

Specifies the technique based on which paths will be made feasible. By default, **set_multicycle_path** technique is used. Other available techniques to make the paths feasible are **path_margin** and **false_path**.

-type setup | **hold** | **both**

Specifies the type of constraints to be written into the timing constraint files. By default, the **setup** constraints are written into these files. Other available options include writing only the **hold** or writing **both** setup and hold constraints into these files. Note that the setup and hold constraints are written only if the respective flags are set true for the scenarios.

-effort medium | **high**

Specifies the effort to be applied to generate the constraints. By default, the effort is **medium** which will generate the constraints of open-ended type. This is very fast, but less accurate. A more accurate effort type is **high** which will generate a more specific close-ended timing constraint. Note that the **high** effort is more runtime intensive but is more accurate.

-apply

Applies the timing constraints to the current design to make the timing paths feasible. If this option is not specified, the command only generates the timing constraint files.

-compress none | **gzip**

Specifies the compression type to use when generating the timing constraint files. By default, no compression is performed.

DESCRIPTION

The **write_feasibility_constraints** command generates constraint files. The constraints can be applied to the current design by using the *-apply* option to convert infeasible timing paths to feasible paths. A timing path having negative slack at the data-pin of an endpoint is said to be an infeasible timing path. The number of constraint files generated is dependent on the total number of modes present in the design.

The **-output** option is required to specify the directory in which to write the constraints. By default, **setup** constraints are generated using **multicycle_path** technique with a feasibility factor of 2.0. Only hold or both type of constraints can be generated using other techniques like **path_margin** and **false_path**; you must specify an appropriate value for *feasibility_factor*.

The actual value of the *feasibility_factor* is calculated based on the clock period and can be selected by the user. If slack of an endpoint is **SLACK** and the capture clock period which reaches the same endpoint is **CLOCK_PERIOD**, the tool considers this endpoint as infeasible if and only if

abs(SLACK) > (feasibility_factor * CLOCK_PERIOD)

If the endpoint is from an IO bound path, then an additional adjustment is made to compute the slack threshold. If the IO constraint value is **IO_CONSTRAINT**, then the tool considers this endpoint as infeasible if and only if

abs(SLACK) > (feasibility_factor * (CLOCK_PERIOD - IO_CONSTRAINT))

The changes made by applying the feasibility constraints can be reverted by using the *remove_feasibility_constraints* command and mentioning the **output** as its input directory.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following example generates Tcl files with timing constraints by using the multicycle_path method and a feasibility factor of 2.0.

```
prompt> write_feasibility_constraints -output Multicycle_Path_2
prompt> all_modes
{Mode_1 Mode_2}
```

The following example writes files with the names Mode_1.tcl and Mode_2.tcl into the MultiCycle_Path_2 directory. The constraints are applied to the current design by the **-apply** switch.

```
prompt> write_feasibility_constraints -output Multicycle_Path_2 -apply
```

The following example generates Tcl files with hold timing constraints by using the false_path method and a feasibility factor of 1.5. The tool writes the files named Mode_1.tcl and Mode_2.tcl to the False_Path_1.5 directory. The **-apply** option can be used to set the constraints for the current design.

```
prompt> write_feasibility_constraints -output False_Path_1.5 \
  -method false_path -feasibility_factor 1.5 -type hold
```

SEE ALSO

remove_feasibility_constraints(2)
set_false_path(2)
set_multicycle_path(2)
set_path_margin(2)

write_floorplan

Writes floorplan information to files.

SYNTAX

status **write_floorplan**
[-include *include_list*]
[-exclude *exclude_list*]
[-output *dir_name*]
[-force]
[-nosplit]
[-format icc | icc2]
[-objects *objects_list*]
[-blocks *blocks_list*]
[-def_units 100 | 200 | 1000 | 2000 | 10000 | 20000]
[-compress gzip | none]
[-add_def_dependencies true | false]
[-via_as_fixed]
[-routed_nets]
[-include_physical_status *include_physical_status_list*]
[-exclude_physical_status *exclude_physical_status_list*]
[-cell_types *cell_types_list*]
[-net_types *net_types_list*]
[-read_def_options *option_string*]
[-def_version *version*]

Data Types

include_list	list
exclude_list	list
dir_name	string
objects_list	collection
blocks_list	collection
include_physical_status_list	list
exclude_physical_status_list	list
cell_types_list	list
net_types_list	list
option_string	string
version	string

ARGUMENTS

-include *include_list*

Specifies the types of data to include in the output. This option cannot be specified together with the **-exclude** or **-objects** options. If this option is not specified, all types are included by default.

The supported types are:

- blockages
- bounds
- cells
- corridors
- die_area
- edit_groups
- io_guides
- macros
- module_boundaries
- nets
- pin_guides
- pins
- route_guides
- rows
- tracks
- vias
- scan_chains
- routing_directions
- voltage_areas
- fills
- pg_metal_fills
- routing_rules
- pg_regions

Specify **-include macros** to output only macro cells and skip standard cells. Specify **-include cells** to output all cells. If you specify **-include {macros cells}**, the command issues a warning message that macros will be ignored.

-exclude *exclude_list*

Specifies the types of data to exclude from the output. This option cannot be specified with the **-include** option or **-objects** options. If this option is specified, the command writes all types, except the types specified in *exclude_list*. The allowed data types are the same as for the **-include** option.

-output *dir_name*

Specifies the name of the directory in which to write the floorplan files. The default is *./floorplan*. If the directory already exists, the

command issues an error message, unless **-force** is also specified.

-force

Overwrites floorplan files in the ./floorplan directory, or in the directory specified by the **-output dir_name** option. By default, the command issues an error message and exits if the directory exists.

-nosplit

Writes out long lines, even if the line length is greater than 80 columns. Use this option to simplify post-processing of the output constraint files or when comparing the constraint files by using the UNIX **diff** command. If this option is not specified, the command breaks long lines near column 80 and inserts a line continuation character (\).

-format icc | icc2

Specifies the target tool for which to write the floorplan. Valid values are **icc** and **icc2**. By default, the value is **icc2**.

To take floorplan information to the Design Compiler tool (in topographical mode) for RTL resynthesis with floorplan information such as macro placement), use the **-format icc** option. The EXAMPLES section contains an example of this flow.

-objects *objects_list*

Specifies the objects to include in the output. Supported object types are cells, ports, blockages, voltage areas, bounds, edit groups, routing corridors, module boundaries, PG regions, site rows, and site arrays. Objects of other types are ignored.

This option cannot be specified with the **-include**, **-exclude**, or **-blocks** options.

-blocks *blocks_list*

Specifies the blocks to output hierarchically. If this option is not specified, only the top-level block is written. This option cannot be combined with the **-objects** option.

-def_units 100 | 200 | 1000 | 2000 | 10000 | 20000

Specifies the number of units per micron in the DEF file. All database length units are scaled accordingly in the output DEF. If not specified, the units per micron is obtained from the technology length precision. Valid values are 100, 200, 1000, 2000, 10000, and 20000. This option is passed to the DEF writer and the behavior is identical to the **-units** option of the *write_def* command.

-compress gzip | none

Specifies the compression type to use when writing the .DEF file. By default, no compression is performed.

-add_def_dependencies true | false

Specifies whether to add dependencies (sections not requested) to the .DEF file to make it self-sufficient. For example, if this option is set to true, if nets are in the **-include** list but vias are not, the VIAS section will be added anyway; if the option is set to false, it will not be added. The default is true.

-via_as_fixed

Indicates that all the via definitions are written out as fixed vias in the output file.

-routed_nets

Indicates that all the nets written out in the output are routed nets, that is, each net has at least one shape associated it.

-include_physical_status *include_physical_status_list*

Specifies which cells to include in the output based on their physical status. When you use this option, all of the specified cells are included in the generated DEF file. The physical status can be one or more of the following:

all

application_fixed
fixed
legalize_only
locked
placed
unplaced

-exclude_physical_status *exclude_physical_status_list*

Specifies which cells to exclude in the output file based on their physical status. When you use this option, all of the specified cell are included in the generated DEF file except for those whose physical status is specified in the *exclude_physical_status_list*.

-include_physical_status and *-exclude_physical_status* are mutually exclusive.

-cell_types *cell_types_list*

Specifies which cells to include in the output DEF file based on their types. When you use this option, all of the specified cells are excluded in the generated file except for those whose cell types is specified in the *cell_types_list*. These are the specifiable cell types:

corner
end_cap
pad
flip_chip_pad
flip_chip_driver
pad_spacer
macro
lib_cell

-net_types *net_types_list*

Specifies which nets to include in the output DEF file based on their types. When you use this option, all of the specified nets are excluded in the generated file except for those whose net types is specified in the *net_types_list*. The net types can be one or more of the following:

clock
ground
power
signal

-read_def_options *option_string*

Specifies options to include with the **read_def** command that is written to the floorplan file.

-def_version *version*

Specifies the DEF syntax version used to create floorplan.def file. Valid values are 5.7 and 5.8. The default is DEF version 5.7.

DESCRIPTION

The **write_floorplan** command writes the *floorplan.tcl* Tcl script and *floorplan.def* DEF file. The *floorplan.def* file generally includes the objects that are numerous, as sourcing the corresponding Tcl would be time-consuming. The *floorplan.tcl* contains the remaining floorplan constructs and a **read_def** command to load the *floorplan.def* file. To restore the floorplan, source the *floorplan.tcl* Tcl script.

By default, the floorplan files are written to a directory named *./floorplan*. The destination can be changed by using the **-output**

option. The command also writes a third file, *floorplan_compare_data.txt*, that contains information to use when comparing the output with a new design by using the **compare_floorplans** command. If the **-blocks** option is used, the command creates subdirectories with names corresponding to the list of blocks.

The **write_floorplan** command writes out user attributes associated with objects to the Tcl script. In the IC Compiler tool, there is no shape or via object associated with a terminal. In this case, the script sets user attributes from the terminal and shape or via object on the terminal for the IC Compiler tool. If there is a conflict between values of the same attribute, a higher precedence is given to the value set on the terminal.

This command returns 1 on success, 0 otherwise.

EXAMPLES

The following example writes floorplan files to the script directory.

```
prompt> write_floorplan -output script
```

The following example writes only cells and blockages.

```
prompt> write_floorplan -include {cells blockages}
```

The following example writes macros hierarchically for all blocks.

```
prompt> write_floorplan -include {macros} -blocks [get_blocks *:*]
```

The following example writes the selected objects and ignores objects that belong to unsupported types.

```
prompt> write_floorplan -objects [get_selection]
```

The following example writes floorplan information in icc format. This could be used to export the floorplan information to the Design Compiler tool (in topographical mode) where the RTL is resynthesized with this floorplan information. Non-PG nets and non-fixed cells are not needed for re-synthesis and so they can be filtered out of the DEF file by using post-processing.

```
prompt> write_floorplan -format icc
```

The following example writes out the floorplan file for all blocks. To load back floorplan file for each block, one method is source the floorplan.tcl Tcl script, the other method is use **set_constraint_mapping_file floorplan/mapfile** and **load_block_constraints -type FLOORPLAN -all_blocks**.

```
prompt> write_floorplan -output floorplan -blocks [lsort -unique \
  [get_attribute [get_cells -hierarchical -filter is_soft_macro] ref_name]
```

The following example writes out the floorplan file and includes the **-add_def_only_objects {cells}** and **-no_incremental** options to the **read_def** command written to the floorplan file.

```
prompt> write_floorplan -output "dirName" -nosplit -verbosity low \
  -def_version 5.8 -read_def_options {-add_def_only_objects {cells} -no_incremental}
```

SEE ALSO

compare_floorplans(2)
read_def(2)
write_def(2)

write_frame_options

Writes a Tcl script that sets the frame view generation options.

SYNTAX

status **write_frame_options**
-library *lib_name*
[-block *block_name*]
[-output *file_name*]
[-format lm_shell | icc2_shell]

Data Types

lib_name string
block_name string
file_name string

ARGUMENTS

-library *lib_name*

Specifies the library for which to output the frame view generation options.

This is a required option.

-block *block_name*

Specifies the blocks for which to output the frame view generation options.

By default, the command writes the frame view generation options for all blocks in the specified reference library.

-output *file_name*

Specifies the output file name.

By default, the frame view generation options are output to the console.

-format lm_shell | icc2_shell

Specifies the output format, lm_shell or icc2_shell.

The default is lm_shell.

DESCRIPTION

This command outputs the original frame view generation options. By default, the command writes the option settings to the console. To save the option settings in a Tcl file, use the **-output** option.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example outputs the frame view generation options for every frame view in the library.

```
prompt> write_frame_options -library testLib \
  -output allOption.tcl
```

The following example outputs the frame view generation options for the specified block in the library.

```
prompt> write_frame_options -library testLib \
  -block testLib/block1/frame \
  -output block1Option.tcl
```

SEE ALSO

create_frame(2)

write_gds

Writes a GDSII stream file from a given design.

SYNTAX

status **write_gds**
[-bus_delimiters *delimiters*]
[-block_map *block_map_file_name*]
[-compress]
[-connect_below_cut_metal]
[-design *design_name*]
[-disable_output_mask_layers *mask_layers*]
[-fill include | exclude | fill_only]
[-flat_vias]
[-foreign]
[-hierarchy top | design | design_lib | all_design_libs | all]
[-ignore_cut_datatype_tbl_mapping]
[-instance_property *instance_property_value*]
[-keep_data_type]
[-layer_map *layer_map_file_name*]
[-layer_map_format icc2 | icc_default | icc_extended]
[-layers *layers*]
[-lib_cell_view frame | layout | design]
[-library *name*]
[-long_names]
[-mask_shifted_suffix *suffix_string*]
[-mask_shifted_suffix_without_constraint *suffix_string*]
[-merge_conflict_suffix *suffix_string*]
[-merge_files *merge_files*]
[-merge_gds_top_cell *top_cell*]
[-merge_cell_shapes]
[-merge_overwrite_conflicting_cell]
[-net_property *net_property_value*]
[-net_text_on all_shapes_and_vias | all_shapes | each_layer | single_shape]
[-output_net_text]
[-output_pin text | geometry | all | none]
[-pin_property *pin_property_value*]
[-via_property *via_property_value*]
[-propagate_pin_mask_to_via_metal]
[-rename_cell *cell_renaming_files*]
[-units *units_per_micron*]
[-report_cell_source *report_file_name*]
[-verbose_report_cell_source*verbose_report_cell_source*]
[-use_block_name]
[-view frame | abstract | timing | layout | design]
[-write_default_layers *layer_collection*]
[-write_instance_shape_mask]

[-write_instance_via_mask]
[-write_instance_blockage_mask]
[-write_instance_mask_on_layers *layers*]
[-exclude_empty_block]
[-allow_design_mismatch]
[-no_marker_layer]
[-switch_view_list *views*]
[-cell_types *cell_type_list*]
[-child_depth *depth*]
[-verbose]
output_file

Data Types

delimiters string
block_map_file_name string
design_name string
mask_layers collection
instance_property_value integer
layer_map_file_name string
layers collection
name string
suffix_string string
merge_files string
top_cell string
net_property_value integer
pin_property_value integer
via_property_value integer
units_per_micron integer
report_file_name string
verbose_report_file_name string
cell_renaming_files string
layer_collection collection
output_file string
views string
cell_type_list list
depth integer

ARGUMENTS

-bus_delimiters *delimiters*

Specifies the bus delimiters to be used in port, pin, and net names to denote bus subscripts. Valid values are "[]", "{}", "()", and "<>".

-block_map *block_map_file_name*

Specifies the name of a block mapping file, a text file containing a list of blocks to ignore (not write out), in the following format:

```
; This is a block mapping file
 myBlockA:ignore     ; ignores myBlockA
 myBlockB:ignore     ; ignores myBlockB
```

-compress

Writes out the GDSII file in gzip (.gz) compressed format to save disk space. A .gz extension is appended to the file name if no extension is specified. By default, the output is uncompressed.

-connect_below_cut_metal

Generates metal layers below the cut metal. For generating metal, it is assumed that two pieces of metal of same size and mask color abut with the cut metal. By default, this action is not performed.

-design *design_name*

Specifies the name of the top design (block) to be written as a GDSII stream. The default is the current block.

-disable_output_mask_layers *mask_layers*

Specifies the layers for which the geometry and text information corresponding to a color-mask configuration are written without considering the color-mask configuration. The color-mask information of the objects is not considered while writing them.

-fill include | exclude | fill_only

Writes or omits fill data according to the specified mode:

- **include**: Writes all geometries, including fill data (default)
- **exclude**: Omits all fill data
- **fill_only**: Writes only the fill data

-flat_vias

Flattens via shapes when writing the GDSII file. By default, vias are written as instances of via masters and are not flattened.

When **-output_net_text** is on and **-net_text_on** is all_shapes_and_vias with this option, the valid text is outputted once, otherwise the text isn't outputted

-force

Writes out the GDSII file, even if the design or library has mismatches; by default, the command issues an error message and stops. This option is deprecated in favor of the **-allow_design_mismatch** option.

-foreign

Uses the LEF foreign cell name as the structure name and instance name. This option affects the STRNAME structure name and the SREF structure element reference name in GDSII. By default, the command ignores the LEF FOREIGN statements and uses the original cell name.

-hierarchy top | design | design_lib | all_design_libs | all

Specifies the parts of the design hierarchy to write out:

- **top**: Writes only the top module in the design, excluding the contents of lower-level modules of the netlist.
- **design**: Writes the design, including lower-level submodules (soft macros), stopping at the design boundary.
- **design_lib**: Writes the design, stopping at its design library boundary.
- **all_design_libs**: Writes the designs in all design libraries but excluding the reference libraries; stops at all design library boundaries.
- **all**: Writes the entire design, including modules in reference libraries.

  The default is "*design_lib*".

-ignore_cut_datatype_tbl_mapping

Ignores the mapping of datatype, based on the via dimensions, from the cutDataTypeTbl entry in the technology file. By default, the command writes cuts of different sizes as different data types as defined by the cut data table in the technology file.

-instance_property ***instance_property_value***

Writes out instance names as a property attribute with the given attribute number in the GDSII stream. By default, no instance property attribute is written.

-keep_data_type

Keeps the data type for each object as defined in the design library (when run in an implementation tool) or the cell library (when run in the library manager). By default, the command converts the data type to 0.

When the **-layer_map** option is used, the layer data type specified in the layer mapping file overrides the data type in the database, and the **-keep_data_type** option setting is ignored. For objects on layers not defined in the layer mapping file, the **-keep_data_type** option still applies.

-layer_map ***layer_map_file_name***

Specifies a layer mapping file that customizes the translation of layer, data type, and purpose between the tool's data model and the GDSII file. For details, see "Layer Mapping File" in the DESCRIPTION section.

-layer_map_format icc2 | icc_default | icc_extended

Specifies the format of the layer mapping file. This option allows the command to accept an IC Compiler layer mapping file. These are the allowed settings:

- **icc_default**: IC Compiler default layer mode, system layers 188-255
- **icc_extended**: IC Compiler extended layer mode, system layers 4001-4095
- **icc2**: IC Compiler II layer mapping file format (default)

-layers ***layers***

Writes only the specified layers of the technology; all other layers are excluded. By default, the command writes all layers to the GDSII file.

-lib_cell_view frame | layout | design

Specifies the name of the view used for writing leaf-level reference library cells to the GDSII file. If the user-specified view is not available for a particular reference library cell, the command skips writing the library cell definition. The default is **layout**.

-library ***name***

Specifies the name of the source library. The default is the current library.

-long_names

Writes out names longer than 32 characters. Use this option to write long hierarchical names without modification. By default, the command limits each name string to 32 characters.

-mask_shifted_suffix ***suffix_string***

Specifies the suffix string used to generate cell master names when the tool performs double-patterning mask swapping. For details, see "Double-Patterning Cell-Level Mask Swapping" in the DESCRIPTION section. This option is mutually exclusive with the **-mask_shifted_suffix_without_constraint** option.

-mask_shifted_suffix_without_constraint ***suffix_string***

Specifies the suffix string used to generate cell master names when the tool performs double-patterning mask swapping. This option differs from the *mask_shifted_suffix* option because the instance-specific mask information is not added to the cell master

names. For details, see "Double-Patterning Cell-Level Mask Swapping" in the DESCRIPTION section.

-merge_conflict_suffix *suffix_string*

Specifies the suffix string to be used to generate new names for resolving merge conflicts that occur with the **-merge_files** option.

-merge_files *merge_files*

Specifies the names of the input stream files to be merged. You can specify both GDSII and OASIS files as input that are to be merged with the newly generated GDSII stream. The command first merges the specified files into an intermediate GDSII file. Then it merges the newly generated GDSII stream with this intermediate file to produce the final GDSII file.

-merge_gds_top_cell *top_cell*

Specifies the top cell to be considered for the merge flow. When this option is specified only the cells in it's hierarchy are considered for merge operation. All remaining cells are removed from the resultant GDSII file.

-merge_cell_shapes

Specifies that the shapes from identically named cells in different source files are to be merged into a single cell in the output file.

-merge_overwrite_conflicting_cell

Writes out the last conflicting cell, only if there are conflicting cells in GDSII files to be merged.

-net_property *net_property_value*

Writes out net names as a property attribute with the given attribute number in the GDSII stream. A property attribute number is an integer from 0 to 127. By default, no net property attribute is written.

-net_text_on all_shapes_and_vias | all_shapes | each_layer | single_shape

Writes or omits net text according to the specified mode:

- **all_shapes_and_vias**: Writes net text on all shapes and vias per net
- **all_shapes**: Writes net text on all shapes (excluding via shapes) per net
- **each_layer**: Writes net text on one shape per layer (excluding via layers) per net
- **single_shape**: Writes net text on one shape per net (default)

  This option require **-output_net_text** to be specified.

-output_net_text

Writes out TEXT objects for net shapes. By default, no TEXT objects are written out for net shapes.

-output_pin text | geometry | all | none

Specifies the manner in which pin objects are written into the GDSII file. Allowed values are:

- "**text**": Writes the name of each pin as a TEXT object associated with the pin.
- "**geometry**": Writes the geometry of each pin as a POLYGON or BOUNDARY object associated with the pin.
- "**all**": Writes both the text and geometry of each pin.
- "**none**": Prevents the writing of pin data.

  The default is "*all*"

-pin_property *pin_property_value*

Writes out pin names as a property attribute with the given attribute number in the GDSII stream. By default, no pin property attribute is written.

-via_property *via_property_value*

Generates the via and via matrix property with the specified attribute number in the GDSII stream. By default, no via or via matrix property is written into GDSII file.

-propagate_pin_mask_to_via_metal

Propagates the double-patterning mask property of each metal1 pin shape, as specified by the pin shape's mask_constraint attribute, to the corresponding metal1 layer of a via dropped on the pin. This modifies the lower_mask_constraint attribute setting of the via from no_mask to match the attribute setting on the pin shape. (An existing setting other than no_mask is not modified).

This option affects only the data written to the GDSII file.

-rename_cell *cell_renaming_files*

Specifies the mapping files which contain pairs of original and new cell names. Each line in the mapping file specifies the name mapping for a single cell.

OldCellName NewCellName

The following example maps ADDFHX1_A to ADDFHX1_B, NOR to NOR2 and XOR to XOR2X4.

#mapfile
ADDFHX1_A ADDFHX1_B
NOR NOR2
XOR XOR2X4

Mask-shifted cells, however, are represented differently in the mapping file. The name mapping comprises three sections: MASKSHIFTLAYER, MASKSHIFTPATTERN and CELLMAP.

For example, to rename the cell A with mask shifting on layer M1 or/and M2, enter the following:

<MASKSHIFTLAYER>
M1 M2
</MASKSHIFTLAYER>
<MASKSHIFTPATTERN>
00 01 10 11
</MASKSHIFTPATTERN>
<CELLMAP>
A AM2B NONE AM1BM2B
</CELLMAP>

MASKSHIFTLAYER specifies the relevant metal layers, in order, on a single row.

MASKSHIFTPATTERN specifies the mask shift combinations to apply to the metal layers. The digits in each combination correspond to the metal layers listed in MASKSHIFTLAYER, respectively. For example, the 00 combination represents a shift value of 0 on layers M1 and M2; the 01 combination represents a shift value of 0 on layer M1 and 1 on layer M2; the 10 combination represents a shift value of 1 on layer M1 and 0 on layer M2; and so on.

The number of mask shift combinations is less than or equal to pow(m, n), where n is the number of layers specified in MASKSHIFTLAYER and m is the count of the shift values. For 2-mask layers, valid shift values are 0 and 1, so m is 2. In the example above, two metal layers are specified (M1 M2), so there are four mask shift combinations (00 01 10 11).

CELLMAP specifies the cell mappings that correspond to each mask shift pattern for a given cell. (Each row specifies the mappings for a single cell, so the number of mappings in each row corresponds to the number of MASKSHIFTPATTERN combinations.) The **write_gds** command renames a cell based on the cell's mask_shift attribute and the mapping file content.

In the example above, cell A is mapped to A, AM2B, NONE, and AM1BM2B when its mask_shift attribute is set to 0 on both layers

(00), set to 1 on layer M2 (01), set to 1 on layer M1 (10), and set to 1 on both layers (11), respectively. When a mask-shifted cell is not expected for a particular mask shift combination, the cell name is mapped to NONE and **write_gds** errors out. When a mask-shifted cell is undefined, the mapping file does not contain the corresponding name mappings and **write_gds** errors out.

By default, the original cell name is preserved in the output GDSII file.

-units ***units_per_micron***

Specifies the resolution to use for writing the GDSII, expressed as the number of units per micron. The default is 10000 (Angstrom units, 1e-10 m).

-report_cell_source ***report_file_name***

Specifies the name of the summary report file in which the physical data of a top-level design is written corresponding to GDSII stream file in plain ascii format. By default, this option writes out the design view of the current block in the current library.For available data, it reports for all the unique libraries used to export cells and one instance from each library which gives user a capability to validate that the exported data come from the right versions of the instantiations be it std cells,IP and blocks. It will write five columns:

- **Path**: The source path or reference location of every reported ndmInstance.
- **Type**: The type of instance whether it is macro_cell,lib_cell,std_cell etc.
- **Reference Name**: The name of the Reference design of that instance.
- **Instance Name**: The name of ndmInst.
- **Physical Hierarchy**: The reference module is present in which all other physical blocks.

-verbose_report_cell_source ***verbose_report_file_name***

Specifies the name of the summary report file.It will report same data as in report_cell_source option but for available data,it reports for all libraries and all physical instances which are exported from them and which are not part of hierarchy of a macro_cell.

-use_block_name

Uses the block name (and user label, if any) instead of the module name for the top structure in the GDSII file. By default, the name used for each structure name is the module name in the data model. In cases where the module name and block name for the top module are different, this option causes the block name to be used; if the block has a user label, the resulting name is *block_user_label*.

-view frame | **abstract** | **timing** | **layout** | **design**

Specifies the name of the view to be written. The default is the **design** view.

-write_default_layers ***layer_collection***

Specifies the layers for which the geometry and text information corresponding to a color-mask configuration is written into the default base. When this option is used with layer-mapping information for the color-masks, all the geometry and text information corresponding to the layers in the input collection are written twice: first into the layer-datatype pair obtained from the layer-map by considering the color-mask configuration, and then again without considering the color-mask configuration.

-write_instance_shape_mask

Writes instance-specific shape masks to the GDSII file as overlapping shapes in the parent structure. For a lib cell instance, the shape mask in bound view will be output, if the value specified by "-lib_cell_view" is different from its bound view. By default, instance-specific shape masks are not written.

-write_instance_via_mask

Writes instance-specific via masks to the GDSII file as overlapping shapes in the parent structure. For a lib cell instance, the via

mask in bound view will be output, if the value specified by "-lib_cell_view" is different from its bound view. By default, instance-specific via masks are not written.

-write_instance_blockage_mask

Writes instance-specific blockage masks to the GDSII file as overlapping shapes in the parent structure. By default, instance-specific blockage masks are not written.

-write_instance_mask_on_layers *layers*

Writes instance-specific mask to the GDSII file on the specified layers only. By default, it is enabled for all the layers.

-allow_design_mismatch

Writes out the GDSII file, even if the design or library has mismatches; by default, the command issues an error message and stops.

-exclude_empty_block

Exclude the empty structure defintions and instances in the GDSII file for each empty block or library cell. By default, the empty block or library-cell defintions are included in the output.

-no_marker_layer

Doesnot generates the marker layer data.

-switch_view_list

Specifies the reference block views for hierarchical block instances to be opened. The view at the head of the list is in higher priority.

-cell_types *cell_type_list*

Writes out only the specified types of cells. By default, the **write_gds** command writes out all types of cells. These are the cell types that you can specify in the *cell_type_list*:

lib_cell
macro
corner
cover
end_cap
filler
flip_chip_driver
flip_chip_pad
pad
pad_spacer
tsv
vib
well_tap

If you use the **-child_depth** option, the **write_gds** command writes out only the types of cells specified by the **-cell_types** option at the current block and their children (including all types of cells in the children) through the specified child depth. It does not write out any other types of cells at the current block nor any of their children.

-child_depth

Specifies the maximum hierarchical block depth to be output. The default value is to output all hierarchical blocks in the hierarchical scope specified by option "-hierarchy". The hierarchical depth is 0 for top block, the depth is increased 1 for child blocks to which instances referred in current block.

-verbose

Prints the details of the hierarchical blocks to be output, and warning messages for those which reference design and view cannot be found.

output_file

Specifies the name of the output GDSII file. If the name ends with the .gz extension, it is compressed using the gzip format.

DESCRIPTION

This command writes out the physical data of a top-level design in a GDSII stream file for tapeout or for export to third-party tools. By default, the command writes out the design view of the current block in the current library and maintains the original layer numbers as it converts the data to GDSII format.

The command offers the following options:

- The name of the library, design, and view to be written out
- The mapping of the database layers to GDSII layers
- The mapping of net, instance, and pin names in the conversion to GDSII format
- The manner of writing out pin objects (text or geometry)
- The handling of mask-shifted multiple-patterning data
- The layers, hierarchical blocks, and fill data to be included in the GDSII output
- The merging of design data with existing GDSII files

To locate files that have relative pathnames, the command uses the **search_path** variable and searches for each file in each directory listed in the **search_path**. Files that have absolute pathnames are loaded as though there were no **search_path**. To determine the file that **write_gds** loads, use the **which** command.

Layer Mapping File

By default, the **write_gds** command maintains the original layer numbers when it writes out the GDSII file.

When there is explicit mapping of system purpose (-1, -2, -3 and -4) in layer mapping file, the tool will use that mapping; for example, with the mapping "all 32:-1 32:55", the tool maps database layer "32:-1" to GDS layer 32:55; if no mapping for these system purposes, the tool will follow the mapping of the purpose 0 of the same layer. For example, if the mapping has "all 32:0 32:70", system purpose "-1" will be mapped to GDS layer 32:70. Please note that GDS datatype can't be any data less than 0, the usage like "all 32:-1 132:-1" is invalid for GDS language, and the datatype will be taken as 0 like "A 32:-1 132 0"

You can optionally provide a layer mapping file to specify the mapping of layers and data types between the database and the GDSII layers. You can use the same mapping file for both the **write_gds** and **read_gds** commands. This is the mapping file format:

```
; single line comment
[read_always { true | false [-mapped_only | -ignore_missing_layers ] } ]
[blockage_as_zero_spacing {true|false}]
[cell_prop_attribute attribute_number]
[net_prop_attribute  attribute_number]
[pin_prop_attribute  attribute_number]
[via_prop_attribute  attribute_number]
[object_type] tool_layer_number[:tool_purpose][:use_type][:mask_type] \
  gdsii_layer_num[:gdsii_data_type]
```

For example,

```
; This is my layer mapping file
                ;
    84      92    ; ICC2 layer 84 = GDSII layer 92
                ;
    59      69:3  ; ICC2 layer 59 = GDSII layer 69, data type 3
                ;
    17:1    117:3  ; ICC2 layer 17, purpose 1 = GDSII layer 117, data type 3
                ;
data 33:0:*:*  1:2  ; ICC2 data on layer 33, purpose 0, any use type,
                ;  any mask type = GDSII layer 1, data type 2
text 33:0:*:*  1:4  ; ICC2 text on layer 33, purpose 0, any use type,
                ;  any mask type = GDSII layer 1, data type 4
                ;
; all other layers and data types match between ICC2 and GDSII
```

In each line, any text after a semicolon "*;*" is a comment.

The **read_always** statement determines how the **read_gds** command handles differences between GDSII layers and database layers. The **blockage_as_zero_spacing** statement determines how the **read_gds** command interprets GDSII blockage layers, as either zero-spacing or design-rule-spacing guides. For details, see the man page for the **read_gds** command.

The **cell_prop_attribute**, **net_prop_attribute**, **pin_prop_attribute**, and **via_prop_attribute** statements specify the property attribute numbers used for storing cell instance, net, pin names, via attribute in the GDSII file. The specified value for the properties should be distinct. For example, the statement "net_prop_attribute 5" causes the **write_gds** command to write out net names of objects as property attribute 5 in the GDSII file. These statements in the layer mapping file have higher priority than the corresponding **write_gds** command options, **-instance_property**, **-net_property**, **-pin_property**, and **-via_property**.

After you write a GDSII file with these properties, when you read the file back into the tool with the **read_gds** command, the original cell names are restored, and the pins and nets are properly identified without the tool spending time on connectivity tracing.

Each field of the layer mapping line is separated by a colon "**:**". Each field can be an explicit value, an asterisk "*****" wildcard character, or left blank. An asterisk or black space indicates any value.

A more specific rule priority than a more general rule. For example, the rule "5:3 8", has higher priority than "5 7". If conflicting rules are specified in different lines (for example, first "6:2 4" and then "6:2 7"), the last one in the file is used and the earlier one is ignored.

The optional *object_type* keyword at the beginning of a layer mapping line restricts the mapping rule to only the specified object types:

- **data**: Mapping applies only to non-text objects
- **text**: Mapping applies only to text objects
- **all**: Mapping applies to both text and non-text objects (default)

tool_layer_number is the layer number in the database, an integer.

tool_purpose is the purpose number in the database, an integer.

use_type is the usage type in the database, which is one of the following.
"route_guide,single_row_via_ladder_pattern_must_join_allowed" requires the relative "tool_layer_number" is any layer number(*), and the format of the layer mapping file is specified as icc_default with -layer_map_format option.
"instance_specific_mask" is to copy the instance-specific mask constraints data (via cut-shapes, shapes, and blockages) to a layer, it is effective when at least one of three options(-write_instance_via_mask, -write_instance_shape_mask and -write_instance_blockage_mask) is specified. Please note, this is "copy" operation, which means the instance specific mask will be written onto original layer still at the same time. By default "power" is to map all power data. "power,net", "power,port", "power,

pin" are to distinguish net, port, pin respectively. The same rules apply to "ground", "clock", "signal" and their corresponding net, port, pin use_types. "same_net_feedthrough_tsv" can be used to create special marking layers in GDS for qualified TSV. Any shape whose mask_name is 'tsv' in design database is duplicated as a marking layer shape in GDS only when "same_net_feedthrough_tsv" is defined in the layer map. The qualified TSV should have a pin. This pin should have a net. The net should have ports of shapes with layer mask_names on backside and front-side both. "hi_voltage" is to index write_gds to output net voltage_range_max as a text. "lo_voltage" is to index write_gds to output net voltage_range_min as a text. The net can be signal, clock, power or ground.

power
power,net
power,port
power,pin
ground
ground,net
ground,port
ground,pin
clock
clock,net
clock,port
clock,pin
signal
signal,net
signal,port
signal,pin
boundary
hard_placement_blockage
soft_placement_blockage
routing_blockage
area_fill
track
route_guide,single_row_via_ladder_pattern_must_join_allowed
bridge_shape
routing_blockage,is_allow_metal_fill_only
instance_specific_mask
same_net_feedthrough_tsv
hi_voltage
lo_voltage
.\"keepout_region

mask_type is the mask type of the shape in a multiple-patterning technology. For a metal layer, the following mask types are supported:

mask_one
mask_two
mask_three
same_mask

For a VIA cut layer, the following additional mask types are supported:

MASK_FOUR
MASK_FIVE
MASK_SIX
...
MASK_FIFTEEN

gdsii_layer_num is the layer number of the geometry or TEXT in the GDSII file, an integer.

gdsii_data_type is the data type number of the geometry or TEXT in the GDSII file, an integer.

Double-Patterning Cell-Level Mask Swapping

In a double-patterning technology, a layer is decomposed into two different masks, with physically adjacent geometries assigned alternately to the two masks. The physical layer is fabricated on the chip using two separate photolithography steps, minimizing optical proximity degradation between adjacent geometries. Double-patterning technology is typically used only for the finest, densest, lowest-level interconnection layers such as M1, VIA1, and M2 layers.

The assignment of adjacent geometries to one mask or the other is called "coloring" and each geometry assigned to a mask is said to have "color" such as mask_one or mask_two. In the cell library, the **mask_constraint** attribute specifies the coloring of each shape in the library cell.

Two shapes of different colors are allowed to be closer to each other than two shapes of the same color. When the spacing between two shapes of the same color is small enough to violate the same-color spacing requirement (but not the different-color spacing requirement), it is called a color violation.

In a "correct-by-construction" standard cell library, the double-patterning shapes are far enough inside the cell boundary to prevent color violations from occurring, even when two cells are placed edge-to-edge. In that case, the tool performs placement and legalization without considering the coloring inside the cells.

On the other hand, for a "precolored" standard cell library, colored shapes are close enough to the cell boundary that color violations can occur between shapes of the same color in two different cells, if the cells are placed edge-to-edge. The placer or legalizer can fix such violations by moving the cells apart in the row, leaving a gap; or possibly by flipping one of the cells in place.

Another way to fix a color violation is to swap the color assignments in the cell instance, without moving or flipping the cell. To do this, the placer sets a cell attribute called mask_shift at the instance level. This attribute is a text string that specifies whether to swap the color assignments within each layer.

For example, when a cell's mask_shift attribute is set to the following string, it swaps the coloring assignments of the shapes in the M1 layer, but not in the VIA1 and M2 layers:

{{M1 1} {VIA1 0} {M2 0}}

Based on the mask_shift attribute setting, the **write_gds** command performs color modification of each cell during stream-out to GDSII format. In this example, the **write_gds** command swaps the coloring assignments in layer M1. For that layer only, it writes out the shapes with the mask_one attribute to the mask_two mask, and conversely, writes out the shapes with the mask_two attribute to the mask_one mask.

The **write_gds** command carries out the modification by writing out an instance of a new master cell master having the switched mask assignments. In effect, the command copies the original cell, modifies the mask assignments as specified by the mask_shift attribute, creates a new master cell having the switched masks, and writes out one or more instances of the new cell.

The **write_gds** command creates a name for the new cell master by joining the following items with underscore characters:

The original cell master name
The suffix string set by the -mask_shifted_suffix option ("SHIFT" by default)
The layer number of first swapped layer
The number 1, indicating the amount to shift the mask for the layer
The layer number of next swapped layer
The number 1, indicating the amount to shift the mask for the layer
...

For example, suppose that the name of the original cell master is INV1, the default suffix string "SHIFT" is being used, and the mask_shift attribute of an INV1 cell instance is set as follows:

{{M1 1} {VIA1 0} {M2 1}}

In that case, the swap occurs in layer M1 (layer 15) and in layer M2 (layer number 17), so the name of the newly generated cell master is "INV1_SHIFT_15_1_17_1".

The tool does not add the new reference cell to the database. It only streams out the new reference cell to the GDSII file.

EXAMPLES

The following command writes out the current block as a GDSII stream file named myDesign.gds:

```
prompt> write_gds -units 1000 myDesign.gds
```

The following command writes out the block myLib:BlockA.design as a GDSII stream file named myBlockA.gds, using a database unit size of 1 nanometer:

```
prompt> write_gds -library myLib -design BlockA -units 1000 myBlockA.gds
```

The following command writes out the current block as a GDSII stream file using a layer mapping file to modify the layers and datatypes, a block mapping file that specifies some blocks to skip, and a cell renaming file to rename some of the blocks before writing:

```
prompt> write_gds -layer_map myLayerMap -block_map myBlockMap \
  -rename_cell myRenamingFile myDesign.gds
```

The following command writes out the current block as a GDSII stream file, including only layers M1, V1, M2, V2, and M3:

```
prompt> write_gds -layers {M1 V1 M2 V2 M3} myDesign.gds
```

SEE ALSO

read_gds(2)
trace_connectivity(2)
search_path(3)

write_io_constraints

Writes signal I/O constraints based on IO placement results to the specified file.

SYNTAX

status **write_io_constraints**
-filename *file_name*
[-format order_only | spacing | pitch | fixed]
[-block]

Data Types

file_name string

ARGUMENTS

-filename *file_name*

Specifies the name of the file to use to write the I/O constraints.

-format order_only | spacing | pitch | fixed

Specifies the I/O constraint format. The default is **order_only**. The **order_only** argument writes only pad order information to the file. The **spacing** argument writes the spacing values between pads and the pad cell names to the file. The **pitch** argument, writes pad cell locations by using cell pitch instead of spacing values. All pad cells must be placed evenly in the IO guide before using the **pitch** argument. The **fixed** argument specifies that all pads cell are fixed at their current locations. The default is **order_only**.

-block

Writes out I/O constraints with respect to the reference blocks in the design. By default, I/O constraints are written from the top level with the hierarchical path to the I/O pad cell.

DESCRIPTION

This command generates signal I/O constraints based on the current IO pad placement. Only pad cells that are legally placed within IO guides are considered. The legal placement is defined as pads that are aligned with IO guides and are in the correct orientation. Pads placed illegally are not included in the constraint file.

The **-filename** option is required. If the **-format** option is not specified, this command generates result by using the **-format order_only** format.

EXAMPLES

The following example writes the IO placement result to the file named io_constraint_file.

```
prompt> write_io_constraints -filename io_constraint_file
```

SEE ALSO

place_io(2)
set_signal_io_constraints(2)

write_ivm

Reads via variation file(.ivm) for current desgin.

SYNTAX

status **write_ivm**
file_name
[-corners *corners*]

Data Types

file_name string
corners list

ARGUMENTS

file_name

Specifies the via variation file name to write.

-corners *corners*

Specifies the list of corners for which to apply the extraction options. If corners are not specified then the ivm file will be applied to all corners.

DESCRIPTION

This command writes .ivm file for via variation analysis for the current design.

EXAMPLES

The following example writes new.ivm file for specified two corners.

prompt> **write_ivm new.ivm -corners {FuncMode_max.SS.081v.125c_FuncCmax.corner FuncMode_min.FF.099v.125c_Func(**

SEE ALSO

report_ivm(2)
remove_ivm(2)
read_ivm(2)
write_parasitics(2)

write_lef

Writes out the data from a library in LEF format.

SYNTAX

string **write_lef**
[-library *library*]
[-design *design*]
[-include *include_list*]
[-properties *property_list*]
[-slice_polygon]
[-write_additional_viarule]
[-version *version_number*]
lef_file_name

Data Types

library	string
design	string
include_list	list
property_list	list
lef_file_name	string
version_number	string

ARGUMENTS

-library *library*

Specifies the library. The default is the current library. This option is mutual exclusive with **-design**.

-design *design*

Specifies the design from which the LEF file will be written. If it has a frame view, it will be written. Otherwise, if it has a design view, all reference designs instanced in it as frame view will be written. This option is mutual exclusive with **-library**.

-include *include_list*

Specifies which sections to include in the LEF output file. By default, all sections are included. The valid settings are:

- **cell**: includes the version, bus bit characters, divider character, macros, and extensions sections
- **tech**: includes all sections except for macros

-properties *property_list*

Specifies which PROPERTY statements to write. Only the properties in the specified list are written; all others are omitted. By

default, all properties are written.

-slice_polygon

Slices the polygons into rectangles for output to the LEF file.

-write_additional_viarule

Specifies the option to write simple via_def as VIARULE.

-version *version_number*

Specifies the version number for the output LEF file. Valid values are 5.6, 5.7 or 5.8. The default is 5.8.

lef_file_name

Specifies the name of the LEF file to be written. If this file already exists, it is overwritten.

DESCRIPTION

This command writes out the data from a library in Library Exchange Format (LEF). By default, the command writes out both the technology information and library cell information.

You can optionally restrict the output to only cell information or only technology information by using the **-include** option. To restrict the types of properties written, use the **-properties** option. To slice polygons into rectangles, use the **-slice_polygon** option.

EXAMPLES

The following example writes a LEF file for library lib_A, including both the technology and library cell information:

```
prompt> write_lef -library lib_A lib_A.lef
```

The following example writes a LEF file from design routed:

```
prompt> write_lef -design design design.lef
```

SEE ALSO

read_lef(2)
read_tech_lef(2)

write_lib_package

Stores a library, its reference libraries, and current application option settings into a package file.

SYNTAX

status **write_lib_package**
[-library *library*]
[-blocks *blocks*]
[-exclude_ref_libs *ref_libs*]
[-exclude_design_view *ref_libs*]
[-verbose]
[-include_all_blocks]
[-include_view *views*]
[-exclude_view *views*]
[-include_label *labels*]
[-exclude_label *labels*]
[-auto_saved_data include | exclude]
[-exclude_dir_path]
file_name

Data Types

library collection
blocks collection
ref_libs collection
file_name string
include_view collection
exclude_view collection
include_label collection
exclude_label collection

ARGUMENTS

-library *library*

The library to store as a package. This can either be a name or a collection of a single library. The default is the current library.

-blocks *blocks*

The collection of blocks to be packed; all other blocks are excluded from the library package.

-include_all_blocks

If this option is used, all blocks in the library are included in the library package. This option is mutually exclusive to **-blocks** option.

-include_view *views*

The views provided with include_view option will be packed in the library package. This option is mutually exclusive to **-include_all_blocks**.

-exclude_view *views*

The views provided with exclude_view option will not be packed in the library package.

-include_label *labels*

The labels provided with include_label option will be packed in the library package. This option is mutually exclusive to **-include_all_blocks**.

-exclude_label *labels*

The labels provided with exclude_label options will not be packed in the library package.

-exclude_ref_libs *ref_libs*

Excludes the specified reference libraries from the library package; only the specified design library (or current library) is included in the package.

Without this option, the **write_lib_package** command includes the reference libraries of the specified (or current) library in the package, and the **read_lib_package** command restores the reference libraries to a new location under the directory of the restored design library.

-exclude_design_view *ref_libs*

Excludes the design views of all the lib_cells from the specified reference libraries. This option does not apply to hierarchical-sub-design design libraries.

-verbose

Reports the command progress and the application options being saved in the package.

-auto_saved_data include | exclude

Specifies whether to include or exclude auto_saved data of the library in the lib_package. By default, auto-saved data will be included.

-exclude_dir_path

If this option is used, all the local directory paths in app-options, ref_libs will be removed before packing.

file_name

The name of the file in which to store the library package.

DESCRIPTION

This command stores a design library, its linked reference libraries, the current application option settings, and the current Tcl variable settings into a library package file.

You can later unpack the file into a self-contained library and restore the tool settings by using the **read_lib_package** command. By default, the reference libraries are restored to a subdirectory below the restored library directory.

To store the design library and exclude one or more reference libraries from the library package file, use the **-exclude_ref_libs**

option of the **write_lib_package** command.

By default, lib_package will be compressed. To disable compression of lib_package, set the app_option **lib.setting.compress_lib_package** to false.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example the packs the current library and its reference libraries into a single file, mylib.pkg.

```
prompt> write_lib_package mylib.pkg
1
```

The following example packs the library named "mylib" into a single file, mylib.pkg. The **-verbose** option reports the progress of the command and the application options and Tcl variables being saved.

```
prompt> write_lib_package -library mylib -verbose mylib.pkg
Packing library mylib...
 App Options
 ...
 Tcl Variables
 ...
Packed 10 app options and 30 tcl variables
Library mylib packed successfully
1
```

The following example packs a specified library into a single file, mylib.pkg. The library is specified as a collection. The library package includes only the "MID" and "BOT" blocks and excludes all reference libraries.

```
prompt> write_lib_package -library [get_libs libA] \
  -blocks [get_blocks {MID BOT}] -exclude_ref_libs * mylib.pkg
1
```

SEE ALSO

read_lib_package(2)
lib.setting.compress_lib_package(3)

write_macro_relative_location

Writes the previously set constraints which place hard macros or macro array relative to anchor objects to file

SYNTAX

status **write_macro_relative_location**
[*target_list*]
[-file *file_name*]
[-hierarchical]

Data Types

target_list list
file_name string

ARGUMENTS

target_list

Specifies the list of hard macros or macro arrays on which to write relative location constraints to file. By default is writing constraints on all hard macros and macro arrays.

-file *file_name*

The file name of macro relative location constraints output. The default is macro_relative_location_constraints.tcl

-hierarchical

This option supports MPH designs to write out macro relative location constraints for hard macros hierarchically.

DESCRIPTION

This command writes hard macro or macro array relative location constraints set by the **set_macro_relative_location** command to file.

EXAMPLES

The following command writes the relative location constraint to file A.tcl.

```
prompt> write_macro_relative_location -file A.tcl
```

SEE ALSO

set_macro_relative_location(2)
remove_macro_relative_location(2)
report_macro_relative_location(2)
derive_macro_relative_location(2)
create_placement(2)
create_macro_relative_location_placement(2)

write_matching_types

Writes out existing assignment of bumps and flip_chip_drivers or user-specified matching types.

SYNTAX

status **write_matching_types**
-file_name *file_name_string*
[-from_existing_assignment]

Data Types

file_name_string string

ARGUMENTS

-file_name *file_name_string*

Specifies the name of the output file with *file_name_string*. This option is required.

-from_existing_assignment

Writes out existing assignment of bumps and flip_chip_drivers, which is the bump assignment results of the **place_io** command. If this option is not specified, the command writes out user-specified matching types.

DESCRIPTION

This command writes out the existing assignment of bumps and flip_chip_drivers, or user-specified matching type assignments, to an output file. Use the output of this command to capture the current assignments; modify and reload the output assignment file as needed.

If you do not specify the **-from_existing_assignment** option, user-defined matching types will retain the names that were assigned by the user.

If you specify the **-from_existing_assignment** option:

a. If only one assignment-based matching type is associated with one user-defined matching type, the new matching type will be named $USER_MATCH_NAME_assignBased, where $USER_MATCH_NAME is the name specified by you and _assignBased is a suffix string.

b. If multiple assignment based matching types are associated with a user matching type, the new matching types will be named $USER_MATCH_NAME_assignBased_$index, where $index is a integer starting from 0.

c. If the assignment-based matching type is not associated with any user matching type, its name will be existing_assignment_$index, where index is a integer starting from 0.

The file name is a required option and can be customized with the **-file_name file_name_string** option.

EXAMPLES

The following example writes out existing assignment to a file named matching_types.tcl.

```
prompt> write_matching_types -file_name matching_types.tcl -from_existing_assignment
```

SEE ALSO

add_to_matching_type(2)
get_matching_types(2)
place_io(2)
remove_from_matching_type(2)
remove_matching_types(2)
report_matching_types(2)

write_name_map

Writes a name map file from the current design.

SYNTAX

string **write_name_map**
name_map_file

Data Types

name_map_file string

ARGUMENTS

name_map_file

Specifies the name of the output name map file. If the file already exists, it is overwritten.

DESCRIPTION

This command writes the name map data from the current block to the specified name map file.

Name map data is loaded into the design using the **read_name_map** command. Note that this data is not automatically modified by netlist editing commands. So this **write_name_map** command can only write out what was originally loaded by the **read_name_map** command.

EXAMPLES

The following example writes a name map file:

prompt> **write_name_map top.nmf**

SEE ALSO

read_name_map(2)

write_net_estimation_rules

Writes a Tcl script that can be used to re-create the net estimation rules.

SYNTAX

int **write_net_estimation_rules**
[-script *script_name*]
[-format tcl | xml]
[-net_estimation_rule *pattern*]
[-cell *cell*]
[-as_cell *cell_name*]
[-include_cell \inc_type]
[rules]

Data Types

script_name string
pattern string
cell collection
cell_name string
inc_type true | false | auto (default)

ARGUMENTS

-script *script_name*

Specifies the name of the script to write. If no script is specified, then the script will be named net_estimation_rules.tcl

-format tcl | xml

Specifies the format of the output script. Valid values are *tcl* or *xml*. The *tcl* format writes a Tcl script that can be sourced by the tool. The *xml* format writes an XML document that can be loaded with the **read_net_estimation_rules** command. If not specified, the default format is *tcl*.

-net_estimation_rule *pattern*

Specifies the net estimation rule to write out.

-cell *cell*

Specifies the physical cell from which to write the net estimation rule.

By default, the current block is searched for the specified net estimation rule(s)..

-as_cell *string*

Specifies the physical cell name to write into the script using -cell. This allows for a quick "copy" of rules from one cell to another. It can take rules from one cell and write them as if they belonged to another. This script can then be read (XML) or sourced (TCL) to put those rules into the new cell.

-include_cell *inc_type*

auto - only include "cell" attribute in script if rule is not in "top" true - always include "cell" attribute in script file false - never include "cell" attribute in script file

rules

Specifies the name of a net estimation rule or rules to report. Net estimation rules are defined with the **set_net_estimation_rule** command. If no rule is specified, then all the rules are written.

DESCRIPTION

This command writes a script that contains rules created with the **set_net_estimation_rule** command.

NET ESTIMATION RULES AND HIERARCHY

Net estimation rules can be created in any physical block at any level of the hierarchy and multiple blocks can have different rules with the same name. However, rules are often stored/retrieved by name, and doing this can cause confusion as to which rule is being used by a given command. Generally, the "current" (top) block's list of rules is searched by commands such as *estimate_timing*. This means that if you define rule R1 in "top" and rule R1 in block "B1", if you make block B1 the current block, commands will use B1's R1. When you go back to top, top's R1 is used. To avoid confusion, use globally unique rule names and define them at the top - they will be propagated to child blocks during distributed commands as necessary. Rules in child blocks that have the same name as rules in top may also be overwritten during distributed commands.

EXAMPLES

The following example creates one non-default rule and write out the script.

```
prompt> set_net_estimation_rule -parameter register_spacing -value 20 myrule
prompt> write_net_estimation_rules
(info) 1 net estimation rules written.
1
```

The following example creates one non-default rule and writes out the rule as an XML document. The output file name is rules.xml.

```
prompt> set_net_estimation_rule -parameter register_spacing -value 20 myrule
prompt> write_net_estimation_rules -script rules.xml -format xml
(info) 1 net estimation rules written.
1
```

The following examples writes rules from cell TOP_CELL/B1 and saves them to an XML file using the "cell" attribute of TOP_CELL/C1.

```
prompt> write_net_estimation_rules -script rules.xml -cell TOP_CELL/B1 -as_cell TOP_CELL/C1 -format xml (info) 1 net estimation rules written.
```

SEE ALSO

read_net_estimation_rules(2)
report_net_estimation_rules(2)
set_net_estimation_rule(2)

write_oasis

Writes an OASIS stream file from a given design.

SYNTAX

status **write_oasis**
[-bus_delimiters *delimiters*]
[-library *name*]
[-design *design_name*]
[-view frame | abstract | timing | layout | design]
[-lib_cell_view frame | layout | design]
[-fill include | exclude | fill_only]
[-layer_map *layer_map_file_name*]
[-layer_map_format icc2 | icc_default | icc_extended]
[-block_map *block_map_file_name*]
[-hierarchy top | design | design_lib | all_design_libs | all]
[-units *units_per_micron*]
[-net_property *net_property_value*]
[-instance_property *instance_property_value*]
[-pin_property *pin_property_value*]
[-via_property *via_property_value*]
[-keep_data_type]
[-compress *compression_level*]
[-output_pin text | geometry | all | none]
[-output_net_text]
[-net_text_on all_shapes_and_vias | all_shapes | each_layer | single_shape]
[-flat_vias]
[-report_cell_source *report_file_name*]
[-verbose_report_cell_source *verbose_report_cell_source*]
[-foreign]
[-ignore_cut_datatype_tbl_mapping]
[-propagate_pin_mask_to_via_metal]
[-write_default_layers *layer_collection*]
[-mask_shifted_suffix *suffix_string*]
[-mask_shifted_suffix_without_constraint *suffix_string*]
[-use_block_name]
[-layers *layers*]
[-write_instance_shape_mask]
[-write_instance_via_mask]
[-connect_below_cut_metal]
[-merge_files *merge_files*]
[-merge_conflict_suffix *suffix_string*]
[-merge_oasis_top_cell *top_cell*]
[-merge_cell_shapes]
[-merge_overwrite_conflicting_cell]
[-rename_cell *cell_renaming_file*]
[-write_instance_blockage_mask]

[-write_instance_mask_on_layers *layers*]
[-allow_design_mismatch]
[-no_marker_layer]
[-switch_view_list *views*]
[-cell_types *cell_type_list*]
[-child_depth *depth*]
[-verbose]
output_file

Data Types

delimiters string
name string
design_name string
layer_map_file_name string
block_map_file_name string
units_per_micron integer
net_property_value integer
instance_property_value integer
pin_property_value integer
via_property_value integer
compression_level integer
report_file_name string
verbose_report_file_name string
layer_collection collection
suffix_string string
layers collection
merge_files string
top_cell string
cell_renaming_file string
views string
output_file string
cell_type_list list
depth integer

ARGUMENTS

-bus_delimiters *delimiters*

Specifies the bus delimiters to be used in port, pin, and net names to denote bus subscripts. Valid values are "[]", "{}", "()", and "<>".

-library *name*

Specifies the name of the source library. The default is the current library.

-design *design_name*

Specifies the name of the top design (block) to be written as an OASIS stream. The default is the current block.

-view frame | **abstract** | **timing** | **layout** | **design**

Specifies the name of the view to be written. The default is the **design** view.

-lib_cell_view frame | **layout** | **design**

Specifies the name of the view used for writing leaf-level reference library cells to the OASIS file. If the user-specified view is not available for a particular reference library cell, the command skips writing the library cell definition. The default is **layout**.

-fill include | exclude | fill_only

Writes or omits fill data according to the specified mode:

- **include**: Writes all geometries, including fill data (default)
- **exclude**: Omits all fill data
- **fill_only**: Writes only the fill data

-layer_map *layer_map_file_name*

Specifies a layer mapping file that customizes the translation of layer, data type, and purpose between the tool's data model and the OASIS file. For details, see "Layer Mapping File" in the DESCRIPTION section.

-layer_map_format icc2 | icc_default | icc_extended

Specifies the format of the layer mapping file. This option allows the command to accept an IC Compiler layer mapping file. These are the allowed settings:

- **icc_default**: IC Compiler default layer mode, system layers 188-255
- **icc_extended**: IC Compiler extended layer mode, system layers 4001-4095
- **icc2**: IC Compiler II layer mapping file format (default)

-block_map *block_map_file_name*

Specifies the name of a block mapping file, a text file containing a list of blocks to ignore (not write out), in the following format:

```
; This is a block mapping file
 myBlockA:ignore     ; ignores myBlockA
 myBlockB:ignore     ; ignores myBlockB
```

-hierarchy top | design | design_lib | all_design_libs | all

Specifies the parts of the design hierarchy to write out:

- **top**: Writes only the top module in the design, excluding the contents of lower-level modules of the netlist.
- **design**: Writes the design, including lower-level submodules (soft macros), stopping at the design boundary.
- **design_lib**: Writes the design, stopping at its design library boundary.
- **all_design_libs**: Writes the designs in all design libraries but excluding the reference libraries; stops at all design library boundaries.
- **all**: Writes the entire design, including modules in reference libraries.

The default is "*design_lib*".

-units *units_per_micron*

Specifies the resolution to use for writing the OASIS data, expressed as the number of units per micron. The default is 10000 (Angstrom units, 1e-10 m).

-net_property *net_property_value*

Writes out net names as a property attribute with the given attribute number in the OASIS stream. A property attribute number is an integer from 0 to 127. By default, no net property attribute is written.

-instance_property ***instance_property_value***

Writes out instance names as a property attribute with the given attribute number in the OASIS stream. By default, no instance property attribute is written.

-pin_property ***pin_property_value***

Writes out pin names as a property attribute with the given attribute number in the OASIS stream. By default, no pin property attribute is written.

-via_property ***via_property_value***

Generates the via and via matrix property with the specified attribute number in the OASIS stream. By default, no via or via matrix property is written into OASIS file.

-keep_data_type

Keeps the data type for each object as defined in the database. By default, the command converts the data type to 0.

When the **-layer_map** option is used, the layer data type specified in the layer mapping file overrides the data type in the database, and the **-keep_data_type** option setting is ignored. For objects on layers not defined in the layer mapping file, the **-keep_data_type** option still applies.

-compress ***compression_level***

Writes out the OASIS file in compressed format to save disk space, using the specified compression level, an integer from 1 to 9. Using a higher number results in greater compression (a smaller file) but slower saving and reading. Using a value of 6 results in a good trade-off between compression and speed. By default, the output is not compressed.

-output_pin text | **geometry** | **all** | **none**

Specifies the manner in which pin objects are written into the OASIS file. Allowed values are:

- "**text**": Writes the name of each pin as a TEXT object associated with the pin.
- "**geometry**": Writes the geometry of each pin as a POLYGON or BOUNDARY object associated with the pin.
- "**all**": Writes both the text and geometry of each pin.
- "**none**": Prevents the writing of pin data.

The default is "*all*"

-output_net_text

Writes out TEXT objects for net shapes. By default, no TEXT objects are written out for net shapes.

-net_text_on all_shapes_and_vias | **all_shapes** | **each_layer** | **single_shape**

Writes or omits net text according to the specified mode:

- **all_shapes_and_vias**: Writes net text on all shapes and vias per net
- **all_shapes**: Writes net text on all shapes (excluding via shapes) per net
- **each_layer**: Writes net text on one shape per layer (excluding via layers) per net
- **single_shape**: Writes net text on one shape per net (default)

  This option require **-output_net_text** to be specified.

-flat_vias

Flattens via shapes when writing the OASIS file. By default, vias are written as instances of via masters and are not flattened.

When **-output_net_text** is on and **-net_text_on** is all_shapes_and_vias with this option, the valid text is outputted once, otherwise the text isn't outputted.

-report_cell_source ***report_file_name***

Specifies the name of the summary report file in which the physical data of a top-level design is written corresponding to GDSII stream file in plain ascii format. By default, this option writes out the design view of the current block in the current library.For available data, it reports for all the unique libraries used to export cells and all instances which gives user a capability to validate that the exported data come from the right versions of the instantiations be it std cells,IP and blocks. It will write five columns:

- **Path**: The source path or reference location of every reported ndmInstance.
- **Type**: The type of instance whether it is macro_cell,lib_cell,std_cell etc.
- **Reference Name**: The name of the Reference design of that instance.
- **Instance Name**: The name of ndmInst.
- **Physical Hierarchy**: The reference module is present in which all other physical blocks.

-verbose_report_cell_source ***verbose_report_file_name***

Specifies the name of the summary report file.It will report same data as in report_cell_source option but for available data,it reports for all libraries and all instances which are exported from them.

-foreign

Uses the LEF foreign cell name as the structure name and instance name. By default, the command ignores the LEF FOREIGN statements and uses the original cell name.

-ignore_cut_datatype_tbl_mapping

Ignores the mapping of datatype, based on the via dimensions, from the cutDataTypeTbl entry in the technology file. By default, the command writes cuts of different sizes as different data types as defined by the cut data table in the technology file.

-propagate_pin_mask_to_via_metal

Propagates the double-patterning mask property of each metal1 pin shape, as specified by the pin shape's mask_constraint attribute, to the corresponding metal1 layer of a via dropped on the pin. This modifies the lower_mask_constraint attribute setting of the via from no_mask to match the attribute setting on the pin shape. (An existing setting other than no_mask is not modified).

This option affects only the data written to the OASIS file, not the vias stored in the database.

-write_default_layers ***layer_collection***

Specifies the layers for which the geometry and text information corresponding to a color-mask configuration is written into the default base. When this option is used with layer-mapping information for the color-masks, all the geometry and text information corresponding to the layers in the input collection are written twice: first into the layer-datatype pair obtained from the layer-map by considering the color-mask configuration, and then again without considering the color-mask configuration.

-mask_shifted_suffix ***suffix_string***

Specifies the suffix string used to generate cell master names when the tool performs double-patterning mask swapping. For details, see "Double-Patterning Cell-Level Mask Swapping" in the DESCRIPTION section. This option is mutually exclusive with the **-mask_shifted_suffix_without_constraint** option.

-mask_shifted_suffix_without_constraint ***suffix_string***

Specifies the suffix string used to generate cell master names when the tool performs double-patterning mask swapping. This option differs from the *mask_shifted_suffix* option because the instance-specific mask information is not added to the cell master

names. For details, see "Double-Patterning Cell-Level Mask Swapping" in the DESCRIPTION section.

-use_block_name

Uses the block name (and user label, if any) instead of the module name for the top structure in the OASIS file. By default, the name used for each structure name is the module name in the data model. In cases where the module name and block name for the top module are different, this option causes the block name to be used; if the block has a user label, the resulting name is *block-name_user-label*.

-layers *layers*

Writes only the specified layers of the technology; all other layers are excluded. By default, the command writes all layers to the OASIS file.

-write_instance_shape_mask

Writes instance-specific shape masks to the OASIS file as overlapping shapes in the parent structure. For a lib cell instance, the shape mask in bound view will be output, if the value specified by "-lib_cell_view" is different from its bound view. By default, instance-specific shape masks are not written.

-write_instance_via_mask

Writes instance-specific via masks to the OASIS file as overlapping shapes in the parent structure. For a lib cell instance, the via mask in bound view will be output, if the value specified by "-lib_cell_view" is different from its bound view. By default, instance-specific via masks are not written.

-connect_below_cut_metal

Generates metal layers below the cut metal. For generating metal, it is assumed that two pieces of metal of same size and mask color abut with the cut metal. By default, this action is not performed.

-force

Writes out the OASIS file, even if the design or library has mismatches; by default, the command issues an error message and stops. This option is deprecated in favor of the **-allow_design_mismatch** option.

-merge_files *merge_files*

Specifies the names of the input stream files to be merged. You can specify both GDSII and OASIS files as input that are to be merged with the newly generated OASIS stream. The command first merges the specified OASIS files into an intermediate OASIS file. Then it merges the newly generated OASIS stream with this intermediate file to produce the final OASIS file.

-merge_conflict_suffix *suffix_string*

Specifies the suffix string to be used to generate new names for resolving merge conflicts that occur with the **-merge_files** option.

-merge_oasis_top_cell *top_cell*

Specifies the top cell to be considered for the merge flow. When this option is specified only the cells in it's hierarchy are considered for merge operation. All remaining cells are removed from the resultant OASIS file.

-merge_cell_shapes

Specifies that the shapes from identically named cells in different source files are to be merged into a single cell in the output file.

-merge_overwrite_conflicting_cell

Writes out the last conflicting cell, only if there are conflicting cells in OASIS files to be merged.

-rename_cell *cell_renaming_file*

Specifies a file that contains the mapping from original cell names to new cell names. The renaming works for both reference

library cells and design blocks. Each line in the file specifies the name mapping for a cell:

OldCellName NewCellName

For example, to map TOP to NEWTOP, enter the following line into the file:

TOP NEWTOP

If only mask shifted cell needs to be renamed, comments with keywords "mask shifted" and "end" need to be added to start and end lines. For example, to map mask shifted cell A to ASHIFT, enter the following lines into the file:

```
; mask shifted cell mapping
A  ASHIFT
; end mask shifted cell mapping
```

By default, the original cell names are written.

-write_instance_blockage_mask

Writes instance-specific blockage masks to the OASIS file as overlapping shapes in the parent structure. By default, instance-specific blockage masks are not written.

-write_instance_mask_on_layers *layers*

Writes instance-specific mask to the OASIS file on the specified layers only. By default, it is enabled for all the layers.

-allow_design_mismatch

Writes out the OASIS file, even if the design or library has mismatches; by default, the command issues an error message and stops. T

-no_marker_layers

Doesnot generates the marker layer data.

-switch_view_list

Specifies the reference block views for hierarchical block instances to be opened. The view at the head of the list is in higher priority.

-cell_types *cell_type_list*

Writes out only the specified types of cells. By default, the **write_oasis** command writes out all types of cells. There are the cell types that you can specify in the *cell_type_list*:

```
lib_cell
macro
corner
cover
end_cap
filler
flip_chip_driver
flip_chip_pad
pad
pad_spacer
tsv
vib
well_tap
```

If you use the **-child_depth** option, the **write_oasis** command writes out only the types of cells specified by the **-cell_types** option at the current block and their children (including all types of cells in the children) through the specified child depth. It does not

write out any other types of cells at the current block nor any of their children.

-child_depth

Specifies the maximum hierarchical block depth to be output. The default value is to output all hierarchical blocks in the hierarchical scope specified by option "-hierarchy". The hierarchical depth is 0 for top block, the depth is increased 1 for child blocks to which instances referred in current block.

-verbose

Prints the details of the hierarchical blocks to be output, and warning messages for those which reference design and view cannot be found

output_file

Specifies the name of the output OASIS file.

DESCRIPTION

This command writes out the physical data of a top-level design in as an OASIS (Open Artwork System Interchange Standard) stream file for tapeout or for export to third-party tools. By default, the command writes out the design view of the current block in the current library and maintains the original layer numbers as it converts the data to OASIS format.

The command offers the following options:

- The name of the library, design, and view to be written out
- The mapping of database layers to OASIS layers
- The mapping of net, instance, and pin names in the conversion to OASIS format
- The manner of writing out pin objects (text or geometry)
- The handling of mask-shifted multiple-patterning data
- The layers, hierarchical blocks, and fill data to be included in the OASIS output
- The merging of design data with existing OASIS files

To locate files that have relative pathnames, the command uses the **search_path** variable and searches for each file in each directory listed in the **search_path**. Files that have absolute pathnames are loaded as though there were no **search_path**. To determine the file that **write_oasis** loads, use the **which** command.

Layer Mapping File

By default, the **write_oasis** command maintains the original layer numbers of the database when it writes out the OASIS file.

When there is explicit mapping of system purpose (-1, -2, -3 and -4) in layer mapping file, the tool will use that mapping; for example, with the mapping "all 32:-1 32:55", the tool maps database layer "32:-1" to oasis layer 32:55; if no mapping for these system purposes, the tool will follow the mapping of the purpose 0 of the same layer. For example, if the mapping has "all 32:0 32:70", system purpose "-1" will be mapped to oasis layer 32:70. Please note that oasis datatype can't be any data less than 0, the usage like "all 32:-1 132:-1" is invalid for oasis language, and the datatype will be taken as 0 like "A 32:-1 132 0"

You can optionally provide a layer mapping file to specify the mapping of layers and data types between the database and the OASIS layers. You can use the same mapping file for both the **write_oasis** and **read_oasis** commands. This is the mapping file format:

```
; single line comment
[read_always { true | false [-mapped_only | -ignore_missing_layers ] } ]
[blockage_as_zero_spacing {true|false}]
[cell_prop_attribute attribute_number]
[net_prop_attribute  attribute_number]
[pin_prop_attribute  attribute_number]
[via_prop_attribute  attribute_number]
[object_type] tool_layer_number[:tool_purpose][:use_type][:mask_type] \
  oasis_layer_num[:oasis_data_type]
```

For example,

```
; This is my layer mapping file
                ;
    84      92    ; ICC2 layer 84 = OASIS layer 92
                ;
    59      69:3  ; ICC2 layer 59 = OASIS layer 69, data type 3
                ;
    17:1    117:3  ; ICC2 layer 17, purpose 1 = OASIS layer 117, data type 3
                ;
data 33:0:*:*  1:2  ; ICC2 data on layer 33, purpose 0, any use type,
                ;  any mask type = OASIS layer 1, data type 2
text 33:0:*:*  1:4  ; ICC2 text on layer 33, purpose 0, any use type,
                ;  any mask type = OASIS layer 1, data type 4
                ;
; all other layers and data types match between ICC2 and OASIS
```

In each line, any text after a semicolon "*;*" is a comment.

The **read_always** statement determines how the **read_oasis** command handles differences between OASIS layers and database layers. The **blockage_as_zero_spacing** statement determines how the **read_oasis** command interprets OASIS blockage layers, as either zero-spacing or design-rule-spacing guides. For details, see the man page for the **read_oasis** command.

The **cell_prop_attribute**, **net_prop_attribute**, **pin_prop_attribute**, and **via_prop_attribute** statements specify the property attribute numbers used for storing cell instance, net, pin names, and via attribute in the OASIS file. The specified value for the properties should be distinct. For example, the statement "net_prop_attribute 5" causes the **write_oasis** command to write out net names of objects as property attribute 5 in the OASIS file. These statements in the layer mapping file have higher priority than the corresponding **write_oasis** command options, **-instance_property**, **-net_property**, **-pin_property**, and **-via_property**.

After you write an OASIS file with these properties, when you read the file back into the tool with the **read_oasis** command, the original cell names are restored, and the pins and nets are properly identified without the tool spending time on connectivity tracing.

Each field of the layer mapping line is separated by a colon "**:**". Each field can be an explicit value, an asterisk "*" wildcard character, or left blank. An asterisk or black space indicates any value.

A more specific rule priority than a more general rule. For example, the rule "5:3 8", has higher priority than "5 7". If conflicting rules are specified in different lines (for example, first "6:2 4" and then "6:2 7"), the last one in the file is used and the earlier one is ignored.

The optional *object_type* keyword at the beginning of a layer mapping line restricts the mapping rule to only the specified object types:

- **data**: Mapping applies only to non-text objects
- **text**: Mapping applies only to text objects
- **all**: Mapping applies to both text and non-text objects (default)

tool_layer_number is the layer number in the database, an integer.

tool_purpose is the purpose number in the database, an integer.

use_type is the usage type in the database, which is one of the following.
"route_guide,single_row_via_ladder_pattern_must_join_allowed" requires the relative "tool_layer_number" is any layer number(*), and the format of the layer mapping file is specified as icc_default with -layer_map_format option.
"instance_specific_mask" is to copy the instance-specific mask constraints data (via cut-shapes, shapes, and blockages) to a layer, it is effective when at least one of three options(-write_instance_via_mask, -write_instance_shape_mask and -write_instance_blockage_mask) is specified. Please note, this is "copy" operation, which means the instance specific mask will be written onto original layer still at the same time. By default "power" is to map all power data. "power,net", "power,port", "power, pin" are to distinguish net, port, pin respectively. The same rules apply to "ground", "clock", "signal" and their corresponding net, port, pin use_types. "same_net_feedthrough_tsv" can be used to create special marking layers in GDS for qualified TSV. Any shape whose mask_name is 'tsv' in design database is duplicated as a marking layer shape in GDS when "same_net_feedthrough_tsv" is defined in the layer map. The qualified TSV should have a pin. This pin should have a net. The net should have ports of shapes with layer mask_names on backside and front-side both. "hi_voltage" is to index write_gds to output net voltage_range_max as a text. "lo_voltage" is to index write_gds to output net voltage_range_min as a text. The net can be signal, clock, power or ground.

power
power,net
power,port
power,pin
ground
ground,net
ground,port
ground,pin
clock
clock,net
clock,port
clock,pin
signal
signal,net
signal,port
signal,pin
boundary
hard_placement_blockage
soft_placement_blockage
routing_blockage
area_fill
track
route_guide,single_row_via_ladder_pattern_must_join_allowed
bridge_shape
routing_blockage,is_allow_metal_fill_only
instance_specific_mask
same_net_feedthrough_tsv
hi_voltage
lo_voltage

mask_type is the mask type of the shape in a multiple-patterning technology. For a metal layer, the following mask types are supported:

mask_one
mask_two
mask_three
same_mask

For a VIA cut layer, the following additional mask types are supported:

MASK_FOUR
MASK_FIVE
MASK_SIX
...
MASK_FIFTEEN

oasis_layer_num is the layer number of the geometry or TEXT in the OASIS file, an integer.

oasis_data_type is the data type number of the geometry or TEXT in the OASIS file, an integer.

Double-Patterning Cell-Level Mask Swapping

In a double-patterning technology, a layer is decomposed into two different masks, with physically adjacent geometries assigned alternately to the two masks. The physical layer is fabricated on the chip using two separate photolithography steps, minimizing optical proximity degradation between adjacent geometries. Double-patterning technology is typically used only for the finest, densest, lowest-level interconnection layers such as M1, VIA1, and M2 layers.

The assignment of adjacent geometries to one mask or the other is called *coloring* and each geometry assigned to a mask is said to have a *color*, such as **mask_one** or **mask_two**. In the database, the **mask_constraint** attribute specifies the coloring of each shape in the library cell.

Two shapes of different colors are allowed to be closer to each other than two shapes of the same color. When the spacing between two shapes of the same color is small enough to violate the same-color spacing requirement (but not the different-color spacing requirement), it is called a color violation.

In a "correct-by-construction" standard cell library, the double-patterning shapes are far enough inside the cell boundary to prevent color violations from occurring, even when two cells are placed edge-to-edge. In that case, the tool performs placement and legalization without considering the coloring inside the cells.

On the other hand, for a "precolored" standard cell library, colored shapes are close enough to the cell boundary that color violations can occur between shapes of the same color in two different cells, if the cells are placed edge-to-edge. The placer or legalizer can fix such violations by moving the cells apart in the row, leaving a gap; or possibly by flipping one of the cells in place.

Another way to fix a color violation is to swap the color assignments in the cell instance, without moving or flipping the cell. To do this, the placer sets a cell attribute called mask_shift at the instance level. This attribute is a text string that specifies whether to swap the color assignments within each layer.

For example, when a cell's mask_shift attribute is set to the following string, it swaps the coloring assignments of the shapes in the M1 layer, but not in the VIA1 and M2 layers:

{{M1 1} {VIA1 0} {M2 0}}

Based on the mask_shift attribute setting, the **write_oasis** command performs color modification of each cell during stream-out to OASIS format. In this example, the **write_oasis** command swaps the coloring assignments in layer M1. For that layer only, it writes out the shapes with the mask_one attribute to the mask_two mask, and conversely, writes out the shapes with the mask_two attribute to the mask_one mask.

The **write_oasis** command carries out the modification by writing out an instance of a new master cell master having the switched mask assignments. In effect, the command copies the original cell, modifies the mask assignments as specified by the mask_shift attribute, creates a new master cell having the switched masks, and writes out one or more instances of the new cell.

The **write_oasis** command creates a name for the new cell master by joining the following items with underscore characters:

The original cell master name
The suffix string set by the -mask_shifted_suffix option ("SHIFT" by default)
The layer number of first swapped layer
The number 1, indicating the amount to shift the mask for the layer
The layer number of next swapped layer

The number 1, indicating the amount to shift the mask for the layer
...

For example, suppose that the name of the original cell master is INV1, the default suffix string "SHIFT" is being used, and the mask_shift attribute of an INV1 cell instance is set as follows:

{{M1 1} {VIA1 0} {M2 1}}

In that case, the swap occurs in layer M1 (layer 15) and in layer M2 (layer number 17), so the name of the newly generated cell master is "INV1_SHIFT_15_1_17_1".

The tool does not add the new reference cell to the database. It only streams out the new reference cell to the OASIS file.

EXAMPLES

The following command writes out the current block as an OASIS stream file named myDesign.oasis:

```
prompt> write_oasis -units 1000 myDesign.oasis
```

The following command writes out the block myLib:BlockA.design as a OASIS stream file named myBlockA.oasis, using a database unit size of 1 nanometer:

```
prompt> write_oasis -library myLib -design BlockA -units 1000 myBlockA.oasis
```

The following command writes out the current block as an OASIS stream file using a layer mapping file to modify the layers and datatypes, a block mapping file that specifies some blocks to skip, and a cell renaming file to rename some of the blocks before writing:

```
prompt> write_oasis -layer_map myLayerMap -block_map myBlockMap \
  -rename_cell myRenamingFile myDesign.oasis
```

The following command writes out the current block as an OASIS stream file, including only layers M1, V1, M2, V2, and M3:

```
prompt> write_oasis -layers {M1 V1 M2 V2 M3} myDesign.oasis
```

SEE ALSO

read_oasis(2)
search_path(3)
write_gds(2)
read_gds(2)
merge_stream(2)

write_optimization_history

Write design optimization history from saved file.

SYNTAX

status **write_optimization_history**
[-file *file_name*]

DESCRIPTION

This command saves the history of the optimization steps applied to the design to a file.

SEE ALSO

read_compile_history(2)
write_compile_history(2)
report_optimization_history(2)

write_parasitics

Writes parasitics to SPEF files for current design.

SYNTAX

status **write_parasitics**
-output <file_name>
[-no_name_mapping]
[-hierarchical]
[-compress]
[-format spef|gpd|attach_gpd]
[-corner <corner_name>]
[-rde_corr]
[-via_area]

Data Types

<file_name> string

ARGUMENTS

-output <file_name>

Specifies the name of the output file to which parasitics for the current design are written.

All corners are written to SPEF files with the following format:

<file_name>.<parasitic_tech_name>_<temperature>[_<user_scaling>].spef

The user scaling is in the following format:

VR: <horizontal_res_scale>_<vertical_res_scale>_<horizontal_cap_scale>_<vertical_cap_sclae>_<via_res_scale>

RDE: <res_scale>_<cap_scale>_<rde_res_scale>_<rde_cap_scale>

Mix-mode/DR: <res_scale>_<cap_scale>_<ccap_scale>

-no_name_mapping

Specifies that the net name is used directly in the file. With no name mapping, the actual net names are used in the SPEF file.

By default, the net names are mapped to numbers that are later used when writing RC details.

-hierarchical

To write hierarchical SPEF files. It generates SPEF files for each block. This switch for the time being doesn't invoke extraction. User should run update_timing before using -hier. If sub-blocks have not been extracted, there will have no sub-blocks SPEF file.

-compress

To write gzip SPEF files.

-format

Specifies the data format to output parasitics, spef, gpd or attach_gpd. If it is not specified, the default parasitics output format is SPEF file. attach_gpd format would attach gpd into ndm design.

-corner <corner_name>

Specifies the corner name to write out SPEF file. It supports for a single corner and SPEF file only.

-rde_corr

To enable RDE RC correlation. When current design is a routed design, by using this option it captures RDE footprint internally from a current design and generate RDE spef files. By comparing normal spef files RC correlation can be done. After generating spef files RDE footprint will be gone.

-rde_corr

Specifies to write out via area information in spef file. When PT uses FC/ICC2 a spef file for via variation spef file needs to have via area information. This option enables to write out via area information in the spef file.

DESCRIPTION

This command writes parasitics for the current design to SPEF files. All corners are written to SPEF files. The file XX.spef_scenario is generated when executing the command and this file lists the scenarios associated with the output spef files. Basically, it honors set_user_units but it only accpets Kohm, ohm and F, pF, fF only.

For up-to-date parasitics, it is recommended to do **update_timing** before **write_parasitics**.

Multicorner-Multimode Support

EXAMPLES

The following example writes parasitics for all corners in the current design.

prompt> **write_parasitics -output design**

prompt> **write_parasitics -output design -format gpd**

SEE ALSO

all_corners(2)
set_extraction_options(2)
report_extraction_options(2)
set_parasitic_parameters(2)

report_parasitic_parameters(2)
update_timing(2)
read_parasitics(2)

write_physical_rules

Writes out abstract physical rules/attributes in PRF format from the target library and all its cells.

SYNTAX

int **write_physical_rules**
[-file *output_file*]
[-include *include_list*]
[-library *lib*]

Data Types

output_file string
lib string
include_list list of string

ARGUMENTS

-file *output_file*:

Specify the file name to be written out. If not specified, the data will be written out to screen.

-include *include_list*

Specifies which rules to be written out.

- **library**: Writes out library level rules.
- **cell**: Writes out cell/pin level rules and attributes.
- **instructions**: Writes out stream in instructions.
- **site**: Writes out site and row_pattern and track_pattern under these two groups.
- **layer**: Writes out layer definition under library level.
- **pin_attr**: Writes out pin level attributes such as "direction","port_type", "pg_type" and "is_secondary_pg". This option can only work together with "cell".
- **routing_blockage**: Writes out routing_blockage definition. This option can only work together with "cell".
- **terminal**: Writes out terminal definition. This option can only work together with "cell".
- **shape**: Writes out shape definition. This option can only work together with "cell".
- **all**: Writes out all rules and attributes.

The default value is *{library cell}*.

-library ***lib***

Specifies the target library to write out the rules and attributes. This option is only avaiable in ICC2. If not specified, current library will be used.

In library manager, it's always the current worksapce library.

DESCRIPTION

This command writes out abstract physical rules and attributes from target library and all its cells in PRF format.

EXAMPLES

The following example writes out rules from library and cells to file out.prf.

```
prompt> write_physical_rules -file out.prf
2
```

The following example writes out instructions and cell level rules/attributes to screen.

```
prompt> write_physical_rules -include {instructions cell}
2
```

SEE ALSO

read_physical_rules(2)
remove_physical_rules(2)

write_pin_constraints

Writes topological and physical pin constraints to a file.

SYNTAX

status **write_pin_constraints**
[-from_existing_pins]
[-pins *pins*]
[-nets *nets*]
[-exclude_nets *nets*]
[-ports *ports*]
[-bundles *bundles*]
[-bundle_pin_constraint]
[-cells *cells*]
[-self]
[-exclude_leaf_cells]
[-topological_map *map_constraint*]
[-physical_pin_constraint *pin_constraint*]
-file_name *filename*

Data Types

pins collection
nets collection
ports collection
bundles collection
cells collection
map_constraint string
pin_constraint string
filename string

ARGUMENTS

-from_existing_pins

Writes the pin constraint file based on the actual pin locations and net connections for the pins in the design, instead of the constraints for the pins. If this option is specified without the **-topological_map** and **-physical_pin_constraint** options, the command writes out the physical pin constraints with side, layer, offset, length and width. If this option is not specified, the command writes out pin constraints based on the existing pin constraint settings.

When this option is used together with the **-cells** option, the command writes out the constraints for the pins of the specified cells.

When this option is used together with the **-self** option, the command writes out the constraints for the top-level ports.

-pins *pin*

Specifies which pins to write out to the constraint file. If this option is not specified, the command writes out constraints for all constrained pins.

-nets ***net***

Specifies which nets to write out to the constraint file. If this option is not specified, the command writes out constraints for all constrained nets.

-exclude_nets ***net***

Specifies the nets whose topological and physical constraints will be skipped when writing the constraint file. The physical pin constraints associated with pins and ports on excluded nets will not be skipped.

If used with the **-from_existing_pins** and **-topological_map** options, the command skips the flat nets that the excluded nets belong to.

If used without option **-from_existing_pins**, the command skips the topological constraints that are associated with the excluded nets.

-ports ***port***

Specifies which ports to write out to the constraint file. If this option is not specified, the command writes out constraints for all constrained ports.

-bundles ***bundle***

Specifies which bundles to write out to the constraint file. If this option is not specified, the command writes out constraints for all constrained bundles. Only top-level bundles can be specified.

-bundle_pin_constraint

Writes bundle pin constraints to the constraint file. Only top-level bundles can be written out.

-cells ***cell***

Specifies which block cell pin constraints to write out to the constraint file. This option is mutually exclusive with the **-nets**, **-pins** and **-ports** options. If this option is not specified, the command writes out constraints for all constrained block cells. Only top-level cells can be specified.

-self

Writes out constraints for the top-level pins. This option is mutually exclusive with the **-nets**, **-pins**, **-ports** and **-cells** options. If this option is not specified, the command writes out the constraints for the blocks in the design.

-exclude_leaf_cells

Excludes the topological constraint segment that contains standard cell pins in output file, as long as the other element in that segment is specified within other topological constraint segments, so that no information on physical blocks or physical pins is lost.

For example, when the option is not used, the topological constraints are:

```
start topological map;
{net A}
  {{pin INV1/Y} {{cell UA} {side 1}}}
  {{{cell UA} {side 1}} {{cell UA/UB} {side 1}}}
  {{{cell UA/UB} {side 3}} {{cell UA} {side 3}}}
  {{{cell UA} {side 3}} {pin INV2/A}};
end topological map;
```

With this option, the segments with standard cell pins will be removed and the output is as follows:

```
start topological map;
{net A}
 {{{cell UA} {side 1}} {{cell UA/UB} {side 1}}}
 {{{cell UA/UB} {side 3}} {{cell UA} {side 3}}};
end topological map;
```

This option can only be used together with the **-from_existing_pins** and **-topological_map** options.

-topological_map ***map_constraint***

Writes out the topological map constraints for the design. This option can write out both top-level and block-level topological constraints.

Specify one or more of the following arguments to write these constraints to the file:

- **side**

  The side number is a positive integer that starts from 1. Given any block with a rectangular or rectilinear shape, the left-most edge is side number 1. If there are multiple left-most edges, then edge 1 is the lowest left-most edge. The sides are numbered in consecutive order proceeding clockwise around the shape. Shape refers to the shape of the block instance, not the shape of the reference block.

- **layer**

  The layer constraint specifies the metal layers to use for the pins.

- **offset**

  The offset specifies the distance in microns from the starting point of a given edge to the routing cross-point on that edge in the clockwise direction. The starting point for edge number 1 is the lower vertex, the starting point for edge number 2 is the left vertex, and so on. You must specify the side number when you specify an offset.

- **offset_range** *real*

  When you specify this argument, the command adds the specified value to, and subtracts it from, the current offset position. For example, if the current offset is 20 and you specify **-topological_map {offset offset_range 5}**, the command writes the constraint as {offset {15 25}}. If the specified value is outside of the boundary, the edge boundary endpoint is used.

  The **offset_range** argument must be zero or a positive number.

- layer_range *int*

  When you specify this argument, the command adds layers above and below the current pin layer to create the output constraint. For example, if the current layer for the pin is M4 and you specify **-topological_map {layer layer_range 2}**, the command adds two layers above and two layers below the current pin layer and writes the constraint as {layers {METAL2 METAL3 METAL4 METAL5 METAL6}}.

  The **layer_range** argument must be zero or a positive number.

When you specify multiple arguments, you can separate them with the vertical bar character (|). The **-topological_map {side | layer | offset}** option and **-topological_map {side layer offset}** option specify the same information.

The command writes out the constraints for the specified arguments only when you also specify the **-from_existing_pins** option. If you do not specify the **-from_existing_pins** option, the command writes out all pin constraints associated with the specified objects stored in design database and ignores the arguments to the **-topological_map** option.

If you specify this option without an argument, the command writes only cell information.

If you do not specify this option, the command writes out all options with no offset range or layer range.

Use this option in an incremental pin-placement flow to write the current constraints, edit them, and read them into the tool.

If no argument is passed to this option and also specified together with -from_existing_pins, then the command will print only the cell information.

The following example writes out offsets with range of 2 microns and layers with a range of 2.

```
write_pin_constraints -file_name pinconstraints.txt -from_existing_pins \
  -topological_map {side | layer | offset | offset_range 2.0 | layer_range 2}
```

If net A is connected from block cell B through side 3 at offset 14 and layer M4, and to cell C through side 1 at offset 24 and layer M4, the **write_pin_constraints** command output is as follows:

```
start topological map;
{nets A}
 {{{cell B} {sides 3}
  {offset {12 16}} {layers {METAL2 METAL3 METAL4 METAL5 METAL6}}}
 {{cell C} {sides 1} {offset {22 26}}
  {layers {METAL2 METAL3 METAL4 METAL5 METAL6}}}};
end topological map;
```

-physical_pin_constraint ***pin_constraint***

Writes out the physical pin constraints for the design.

Specify one or more of the following arguments to write these constraints to the file:

- **side**

  The side number is a positive integer that starts from 1. Given any block with a rectangular or rectilinear shape, the left-most edge is side number 1. If there are multiple left-most edges, then edge 1 is the lowest left-most edge. The sides are numbered in consecutive order proceeded clockwise around the shape. The shape here referring to reference block's shape, not the block instance's shape.

- **layer**

- **location**

  If you specify location, the command writes out the coordinate of the center of the pin in the reference block or reference design.

- **offset**

  If you specify offset, the command writes out the offset calculated based on the center of the corresponding pin.

  The offset specifies the distance in microns from the starting point of a given edge to the routing cross-point on that edge in the clockwise direction. The starting point for edge number 1 is the lower vertex, the starting point for edge number 2 is the left vertex, and so on. You must specify the side number when you specify an offset.

- **width**

- **length**

- **order**

  The order specifies the pin constraints start from the pin placed nearest to the starting point of a particular side and sorted based on their distance from the starting point.

- **offset_range** *real*

  When you specify this argument, the command adds the specified value to, and subtracts it from, the current offset position. For example, if the current offset is 20 and you specify **-physical_pin_constraint {offset offset_range 5}**, the command writes the constraint as {offset {15 25}}. If the specified value is outside of the boundary, the edge boundary endpoint is

used.

- **layer_range** *int*

  When you specify this argument, the command writes out the layer range using the layer for the current pin as the middle layer.

When you specify multiple arguments, you can separate them with the vertical bar character (|). The **-physical_pin_constraint {side | layer | offset}** option and **-physical_pin_constraint {side layer offset}** option specify the same information.

The command writes out the constraints for the specified arguments only when you also specify the **-from_existing_pins** option. If you do not specify the **-from_existing_pins** option, the command writes out all pin constraints associated with the specified objects stored in design database and ignores the arguments to the **-physical_pin_constraint** option.

If you do not specify this option, the command writes out all options with no offset range or layer range.

The following example output is written out when you specify the option **-physical_pin_constraints {side | offset | layer}**.

```
start physical pin constraint;
{pins pin_name} {reference ref_block_name} {sides 1} {offset 14} {layers metal1};
.....
end physical pin constraints;
```

-file_name ***filename***

Specifies the file name for the pin constraints output file.

DESCRIPTION

This command writes out the routing topology of nets that are connected to block pins. It can also write out physical information of existing block pins or from existing pin constraints.

The **-pins**, **-nets**, **-ports**, **-cells** and **-self** options are mutually exclusive, you can specify only one of these options with the **write_pin_constraints** command.

If none of the **-topological_map**, **-physical_pin_constraint**, **-cells** and **-self** options are specified, the command writes the topological map section, physical constraints section, pin spacing per edge per layer and block pin constraints sections.

You can run this command multiple times to write out different constraints to different files and specify different objects (nets, pins or cells). The separate files can be concatenated to create a constraint file with multiple topological map sections or multiple physical pin constraint sections. If the same object is written out multiple times with same constraints, the last record takes precedence. If the same object is written out multiple times with different constraints, they are combined together. If the topological map section conflicts with the physical pin constraints section, the topological map section takes precedence. This command only works in design view. If current design is in abstract view, users need to merge data to design view first.

EXAMPLES

The following example writes topological pin constraints to the topo.cstr file based on existing pin placement. The command writes the side, layer, and offset constraints.

```
prompt> write_pin_constraints -from_existing_pins -file_name topo.cstr \
  -topological_map {side | layer | offset}
```

To write all the user-specified individual pin constraints and
topological constraints stored in the design database to pin.cstr file

```
prompt> write_pin_constraints -file_name pins.cstr  -physical_pin_constraint {} \
  -topological_map {}
```

SEE ALSO

place_pins(2)
read_pin_constraints(2)
remove_individual_pin_constraints(2)
report_individual_pin_constraints(2)
set_individual_pin_constraints(2)

write_pt_checksum

This command is for generating checksums for PT.

SYNTAX

status **write_pt_checksum**
[-pt_exec_path *pt_shell_path*]
[-pt_user_script *pt_user_script*]
[-output_directory *output_directory_path*]
[-type *checksum_type_list*]

Data Types

pt_shell_path string
the_script string
output_directory_path string
checksum_type_list list

ARGUMENTS

-pt_exec_path *pt_shell_path*

The path to the PT executable.

-pt_user_script *the_script*

User specified script for PT session.

-output_directory *output_directory_path*

Directory in which checksums will get generated.

-type *checksum_type_list*

Specifies the list of checksums to be generated.

Valid values it can take are **all**, **library**, **design**, **parasitics**, and **constraints**.

If this option is not specified, all checksums are generated.

DESCRIPTION

The **write_pt_checksum** command generates checksums for a pt session.

The **write_pt_checksum** command generates the checksums for a pt session. The command generates two log files pt_checksum.log and pt_output.log. If -type option is not specified all the checksums are generated. The values it can take are **all**, **library**, **design**, **parasitics**, and **constraints**.

EXAMPLES

The following example generates checksums in checksum_out directory.

```
prompt> write_pt_checksum -pt_exec_path $PT_HOME/bin/pt_shell \
  -pt_user_script pt_user_script.tcl -output_directory checksum_out \
  -type {library design}
```

SEE ALSO

write_checksum(2)
compare_checksum(2)

write_push_down_eco

Generate an eco script that can move down a tree of std-cells, from top physical level to the overlapping blocks on next physical level.

SYNTAX

string **write_push_down_eco**
  [-dir *dir_name*]
  \fcell_list

Data Types

dir_name string
cell_list list

ARGUMENTS

-dir *dir_name*

Specifies the directory to output the eco files. The default directory is *psuh_down_eco_scripts*. One eco file will be generated for each block. User is supposed to source the top block script, which will call the scripts for other blocks.

cell_list

Specifies the cells to be pushded down.

DESCRIPTION

This command will generate an eco script that can move down a tree of std-cells, from top physical level to the overlapping blocks on next physical level.

EXAMPLES

prompt> **write_push_down_eco [get_cells U0]**
prompt> **source ./push_down_eco_scripts/DUT.tcl**

write_rde

Writes RDE model to a encrypted file for current design.

SYNTAX

status **write_rde**
file_name
[-help]

Data Types

file_name string

ARGUMENTS

file_name

Specifies the name of the output file to which a RDE model for the current design are written.

-help

Specifies the usage of write_rde

DESCRIPTION

This command writes RDE model for the current design to an encrypted file. For very large designs, the runtime of the RDE capture step can be reduced by dumping the RDE model from a prior run with "write_rde" and reading it back into a successive run with "read_rde". Once the RDE model information is read in, before either the place_opt or clock_opt flow stages, the subsequent RDE capture step(s) will be skipped subject to the following use model. A RDE model dumped after place_opt (final_opto) may be reused only by the place_opt call of the successive run. An RDE model dumped after clock_opt may be used in both the place_opt and clock_opt stages of the new run (the model need only be read in one time, before either place_opt or clock_opt). The "read_rde" command includes some checking as to whether there have been floorplan changes and, if so, may not allow the RDE model to be read in. In this case, the user should run the default flow and regenerate new RDE model information after the RDE capture stages if desired.

USAGE

run 1 : place_opt -from final_opto write_rde place_opt_RDE ... clock_opt -from final_opto write_rde clock_opt_RDE

run 2 : read_rde place_opt_RDE OR read_rde clock_opt_RDE place_opt -from final_opto ... read_rde clock_opt_RDE clock_opt -from final_opto

EXAMPLES

The following example writes a RDE model for the current design.

```
prompt> write_rde -help
Usage: write_rde    # a RDE stream out command
       [file_name]          (filename to write out)
prompt> write_rde my.rde
```

SEE ALSO

read_rde(2)

write_routes

Writes the routing information to the specified file.

SYNTAX

status **write_routes**
-output *file_name*
[-nets *collection_of_nets*]
[-objects *collection_of_objects*]

Data Types

output string
nets collection
objects collection

ARGUMENTS

-output *file_name*

Specifies the output file name. Routing information is written to *file_name*. This is a required argument.

-nets *collection*

Specifies the collection of nets for which to write out routing information. If you specify more than one net, separate them with a space.

-objects *collection*

Specifies a collection of shapes and/or vias.

The **-nets** and **-objects** are mutually exclusive.

If you do not use **-nets** nor **-objects**, the tool writes out all routing information by default.

DESCRIPTION

The **write_routes** command outputs the routing information as a set of tcl commands to the specified file. Users can source the file back into the tool.

Limitations

Shielding association is not supported.

EXAMPLES

The following examples write the routing information to a file called my_route.txt.

prompt> **write_routes -output my_route.txt**
prompt> **write_routes -objects [get_selection] -output my_route.txt**

SEE ALSO

write_def(2)
create_shape(2)
create_via(2)

write_routing_constraints

Writes routing constraints of the design in a Tcl script.

SYNTAX

status **write_routing_constraints**
file_name
[-significant_digits *digits*]

Data Types

file_name string
significant_digits int

ARGUMENTS

file_name

The generated Tcl script name.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default value depends on the tech precision. Use this option if you want to over-ride the default.

RETURN VALUE

Returns 1 if the Tcl script is successfully write out, and 0 if not.

DESCRIPTION

The **write_routing_constraints** command writes the routing constraints in the current design in form of a Tcl script.

The routing constraints which will be written by the command are create_routing_rule, set_routing_rule, create_wire_matching, create_length_limit, create_differential_group, create_net_shielding, create_net_priority, create_bus_routing_style and set_ignored_layers.

EXAMPLES

The following example writes routing constraints of the current design in **file_name** file.tcl:

```
prompt> write_routing_constraints file
1
```

SEE ALSO

create_routing_rule(2)
set_routing_rule(2)
create_wire_matching(2)
create_length_limit(2)
create_differential_group(2)
create_net_shielding(2)
create_net_priority(2)
create_bus_routing_style(2)
set_ignored_layers(2)

write_rp_groups

Writes out the relative placement constraints for the specified relative placement groups.

SYNTAX

int **write_rp_groups**
rp_group_list [-hierarchical] | -all
[-nosplit]
[-file_name *filename*]
[-create]
[-cell]
[-blockage]
[-rp_group]

Data Types

rp_group_list list or collection
filename string

ARGUMENTS

rp_group_list

Specifies the relative placement groups for which constraints have to be written out.

This option is mutually exclusive with the **-all** option.

-all

Specifies that constraints for all relative placement groups of current block are to be written out. Please note that constraints of relative placement groups of child blocks are not written.

This option cannot be used with either the *rp_group_list* argument or the **-hierarchical** option.

-hierarchical

Specifies that constraints for all relative placement groups within the hierarchy of the groups in *rp_group_list* are to be written out. By default, subgroups are not written out.

This option is mutually exclusive with the **-all** option.

-nosplit

Indicates that lines are not to be split when column fields overflow. This is most useful when comparing previous script files with the UNIX diff command, or for post-processing the script.

-file_name *filename*

Specifies the name of the file to which the relative placement constraints will be written. The default is to write to standard output.

-create

Specifies that **create_rp_group** and **set_rp_group_options** commands should be written out.

-cell

Specifies that **add_to_rp_group -cell** commands should be written out.

-blockage

Specifies that **add_to_rp_group -blockage** commands should be written out.

-rp_group

Specifies that **add_to_rp_group -rp_group** commands should be written out.

DESCRIPTION

The **write_rp_groups** command writes the relative placement constraints for the specified relative placement groups. You can use the generated commands to re-create the relative placement group constrains on the same design. Please note that constraints can be sourced back only from the block they are written from.

When you specify any of the **-create**, **-cell**, **-blockage**, and **-rp_group** options, the generated constraints contains only the commands related to the specified options. If you do not specify any of these options, all commands are written out.

EXAMPLES

The following example uses **write_rp_groups** to re-create relative placement groups after their removal.

```
prompt> get_rp_groups
{grp_mul grp_ripple top_group}

prompt> write_rp_groups -all -file_name my_groups.tcl
1

prompt> remove_rp_groups -all
3

prompt> source my_groups.tcl
{grp_mul grp_ripple top_group}

prompt> get_rp_groups
{grp_mul grp_ripple top_group}
```

SEE ALSO

add_to_rp_group(2)
create_rp_group(2)
remove_rp_groups(2)

write_safety_register_script

Writes a script file for creating safety register rules and groups.

SYNTAX

status **write_safety_register_script**
-output *filename*

Data Types

filename string

ARGUMENTS

-output *filename*

The name of the output text file where the script for generating rules and groups will be written out. If the file already exists, it will be overwritten.

DESCRIPTION

Writes out a Tcl script file containing commands for creating the safety_register_rules and safety_register_groups. This information can be used in tools in downward flow which do not accept the same database format. The file will be written in ASCII text format and will contain the commands like create_safety_register_rule and create_safety_register_group.

EXAMPLES

The following example writes out a script file which can be sourced in another tool to re-create the safety data.

prompt> **write_safety_register_script -output ./create_safety_data.tcl**

SEE ALSO

create_safety_register_rule(2)

set_safety_register_rule(2)
create_safety_register_group(2)

write_saif

Writes a backward Switching Activity Interchange Format (SAIF) file.

SYNTAX

status **write_saif**
file_name
[-duration *duration_value*]
[-cells *cell_list*]
[-no_hierarchy]
[-propagated]
[-switching_activity_types *act_list*]
[-exclude_sdpd]
[-scenarios *scenario_list*]
[-modes *mode_list*]
[-corners *corner_list*]
[-compress gzip | none]

Data Types

file_name string
duration_value float
cell_list list
act_list list
scenario_list list
mode_list list
corner_list list

ARGUMENTS

output_file_name

Specifies the name of the SAIF file to write.

-duration *duration_value*

Specifies the duration value as a floating point number. The reported power numbers depend on the duration value specified. The unit of the duration is always assumed to be in nanosecond. If not specified, the duration is assumed to be 1.0 ns.

-cells *cell_list*

Specifies the cell list for which the SAIF file is written. The SAIF file has the activity information for the top level of the design. It prints out the switching activity of the instances present down the hierarchy ONLY if the cell_list contains the whole hierarchy of the instance.For an instance U1/U2/U3 the cell_list has to be {U1 U1/U2 U1/U2/U3}.

-no_hierarchy

Makes the command write out SAIF information non-hierarchically. If not used, the default behavior is to write out SAIF hierarchically till the leaf cell.

-propagated

Runs the propagation engine before writing out SAIF file. The effort level for propagation is taken from the value specified by the propagation effort app option. The default level of propagation is low. After running the propagation engine the SAIF file is written out.

-switching_activity_types *act_list*

Specifies the switching activity types for which the SAIF file is written out. The list can have values - annotated / simulated / propagated / sta / default. If this option is not used the annotated and simulated information is written out. This option also supports the value 'all'. If 'all' is used with this option then all types of switching activities are written out.

-exclude_sdpd

Inhibits the state-dependent and path-dependent (SDPD) switching activity generation. If you do not specify this option, the **write_saif** command outputs SDPD information by default. When the **write_saif** command is called with the *-propagated* option, nonannotated SDPD information is also propagated and generated in the SAIF file. *-propagated* and *-exclude_sdpd* options can be used together to output the propagated simple switching activity and no SDPD information.

-scenarios *scenario_list*

Specifies the scenarios to be considered. If this option is not specified, then only those scenarios are considered which contain the modes and corners specified through **-modes** and **-corners** options. If none of the **-modes**, **-corners**, or **-scenarios** options are specified, SAIF file is written for the current scenario in the design.

If the filtered active scenario name is scn1 and the output file name is file1 then the SAIF file written out has the name file1_scn1. Similarly files are written for each scenario if there are more than 1 filtered active scenarios. If there is only 1 scenario then the output SAIF file has name file1.

-modes *mode_list*

Specifies the mode filter on the set of scenarios to be considered. A scenario is considered if its mode is present in *mode_list*, and is active. The criteria for a scenario to be filtered is only based on the active flag set on a scenario and does not depend on the dynamic/leakage power flags. If this option is not specified, scenarios are not filtered based on modes.

-corners *corner_list*

Specifies the corner filter on the set of scenarios to be considered. A scenario is considered if its corner is present in *corner_list*, and is active. The criteria for a scenario to be filtered is only based on the active flag set on a scenario and does not depend on the dynamic/leakage power flags. If this option is not specified, scenarios are not filtered based on corners.

-compress gzip | none

This option takes one of the two values 'gzip' or 'none'. 'gzip' compresses the output SAIF file. 'none' does not compress the output SAIF file. If this option is not specified then by default the output SAIF file is not compressed.

DESCRIPTION

The **write_saif** command writes a backward SAIF file. The default behavior is to write out the annotated or simulated switching activity information including state-dependent and path-dependent for the complete design without running the propagation engine. This command only works on the valid active scenarios. Even if the leakage/dynamic power flags for a scenario are enabled/set ,but the active flag for a scenario is false/not-set, the scenario is not considered for write_saif. The **'-scenarios', '-modes', '-corners'**

options filter out the active scenarios from all the scenarios present in the design. If these options are not used, then the current scenario is used **if** it is active. If the current scenario is not active and the **'-scenarios', '-modes', '-corners'** options are not specified, no SAIF is written out.

If there are more than 1 filtered active scenarios, the SAIF file written out for each scenario has the name *ouput_file_name_scn_name*. If there is only one filtered active scenario, SAIF file with name *output_file_name* is written out without any suffix.

Multicorner-Multimode Support

EXAMPLES

The following example writes the SAIF file for the current design.

```
prompt> write_saif test.saif
```

The following example writes out the SAIF for the top design, U1 and U1/U2 instances.

```
prompt> write_saif -cells {U1 U1/U2} test.saif
```

The following example runs the propagation and then writes out the SAIF having propagated or annotated activity information.

```
prompt> write_saif -propagated -switching_activity_types {annotated} test.saif
```

The following example writes out the non state-dependent and path-dependent switching activity information non-hierarchically.

```
prompt> write_saif -no_hierarchy test.saif
```

The following example writes out all the types of switching activity information - annotated,simulated,propagated,sta,default.

```
prompt> write_saif -switching_activity_types {all} test.saif
```

The following example writes out SAIF files test.saif_scn1 and test.saif_scn2 for the active scenarios scn1 and scn2 present in the design.

```
prompt> write_saif -scenarios {scn1 scn2} test.saif
```

The following example writes out SAIF file test.saif in compressed format.

```
prompt> write_saif -compress gzip test.saif
```

SEE ALSO

get_switching_activity(2)
read_saif(2)

report_power(2)
report_switching_activity(2)
set_switching_activity(2)

write_sanity_check_point

Writes out a sanity check point for the current design.

SYNTAX

status **write_sanity_check_point**
-stage setup | placement
[-output *file_name*]

Data Types

file_name string

ARGUMENTS

-stage setup | placement

Specifies the different design stage for sanity check. For "setup" stage, the command will write out design sanity check points with zero-interconnection delay setting. For "placement" stage, the command will internally run create_placement, then write out design sanity check point with zero-interconnection delay setting.

-output *file_name*

Specifies the name of the generated sanity check point file. For setup stage, tool will generate sanity_setup_qor_log as default output file. For placement stage, tool will generate sanity_placement_qor_log as default output file.

DESCRIPTION

This command writes out sanity check points for the current design. The sanity check points include information such as design area and buffer information, number of dont-touch cells, number of size-only cells, number of fixed placement cells, settings on ignored layers, path groups, design critical path information, QoR (WNS, TNS, NVE), logical DRC violations, number of placement blockages, number of bounds, number of physical-only cells, utilization, and placement wire length.

Note: Due to engine implementation differences, the tool derived size-only and dont-touch numbers might differ between the tools.

EXAMPLES

The following example writes sanity check points for the design.

```
prompt> write_sanity_check_point -stage setup

Writing Sanity Check Point

Finish Sanity Check

1
```

SEE ALSO

report_design(2)
report_qor(2)
create_placement(2)

write_scan_def

Generates SCANDEF scan chain information for performing scan chain reordering in the physical implementation flow.

SYNTAX

status **write_scan_def**
  [-output *def_file*]
  [-version *def_version*]

Data Types

def_file string
def_version string

ARGUMENTS

-output *def_file*

Specifies the name of the SCANDEF output file to create for the physical implementation tool. The default name is current_design.def, in the current directory.

-version *def_version*

Specifies the DEF version of the output file. Valid values are **5.7** and **5.8**.

DESCRIPTION

The **write_scan_def** command generates scan chain information in SCANDEF format. It is a post-DFT command, intended for use after scan chains are inserted into the design.

EXAMPLES

```
prompt> write_scan_def -output scan.def
```

SEE ALSO

preview_dft(2)
compile(2)
set_scan_configuration(2)
create_test_protocol(2)

write_script

Writes shell commands to save the current settings.

SYNTAX

status **write_script**
[-modes *mode_list*]
[-corners *corner_list*]
[-include *include_list*]
[-exclude *exclude_list*]
[-format dc | icc | icc2| pt]
[-output *dir_name*]
[-force]
[-nosplit]
[-compress gzip]

Data Types

mode_list	list
corner_list	list
include_list	list
exclude_list	list
dir_name	string

ARGUMENTS

-modes *mode_list*

Specifies the modes to write.

By default, all modes are written.

-corners *corner_list*

Specifies the corners to write.

By default, all corners are written.

-include *include_list*

Specifies the types of data to include in the output.

Valid values are **annotation**, **aocvm_coefficient**, **aocvm_coefficient_lib_cell**, **app_options**, **block_to_top_map**, **budget**, **case_analysis**, **cell_mode**, **clock**, **clock_cell_spacing_rule**, **clock_gating_check**, **clock_group**, **clock_jitter**, **clock_latency**, **clock_latency_2**,**clock_max_level_control**,**clock_multisource_options**, **clock_scale_mode_groups**, **clock_sense**, **clock_target_skew_latency**, **clock_transition**, **clock_tree_reference_subset**, **clock_uncertainty**,

create_cell_bus,**crpr_scale_factor**, **cts_balance_point**, **cts_exception**, **cts_extended_constraints**, **cus_clock_arrival**, **data_check**, **data_check_ignore**, **disable_timing**, **dont_touch**, **dont_use**, **drc_disable_flags**, **drc_disable_types**, **drive_info**, **edrc**, **exception**, **file_line_info**, **freeze_ports**, **global_route**, **ideal_network**, **idn_annotation**, **ignored_layers**, **input_delay**, **interclock_balance**, **isolate_ports**, **latch_loop_breaker**, **latency_adjustment**, **lib_pin_noise_margin**, **load**, **max_low_vth**, **max_time_borrow**, **min_pulse_width**, **net_resistance**, **output_delay**, **parasitic_corner**, **parasitic_global**, **path_group**, **path_ignored_block**, **placement_spacing_rule**, **port_pin_noise_margin**, **power_clock_scaling\f, power_derate\f, pvt, pvt_dynamic, routing_rule, scaling_lib_groups, \fscan_chain, size_only, skew_group, target_library_subset,timing_derate, via_ladder_commands and voltage_area_rule.**

By default, all types of data are included in the output.

You cannot specify this option with the **-exclude** option.

-exclude ***exclude_list***

Specifies the types of data to exclude from the output.

Valid values are **annotation**, **aocvm_coefficient**, **aocvm_coefficient_lib_cell**, **app_options**, **block_to_top_map**, **budget**, **case_analysis**, **cell_mode**, **clock**, **clock_cell_spacing_rule**, **clock_gating_check**, **clock_group**, **clock_jitter**, **clock_latency**, **clock_latency_2**,**clock_max_level_control**,**clock_multisource_options**, **clock_scale_mode_groups**, **clock_sense**, **clock_target_skew_latency**, **clock_transition**, **clock_tree_reference_subset**, **clock_uncertainty**, **create_cell_bus**,**crpr_scale_factor**, **cts_balance_point**, **cts_exception**, **cts_extended_constraints**, **cus_clock_arrival**, **data_check**, **data_check_ignore**, **disable_timing**, **dont_touch**, **dont_use**, **drc_disable_flags**, **drc_disable_types**, **drive_info**, **edrc**, **exception**, **file_line_info**, **freeze_ports**, **global_route**, **ideal_network**, **idn_annotation**, **ignored_layers**, **input_delay**, **interclock_balance**, **isolate_ports**, **latch_loop_breaker**, **latency_adjustment**, **lib_pin_noise_margin**, **load**, **max_low_vth**, **max_time_borrow**, **min_pulse_width**, **net_resistance**, **output_delay**, **parasitic_corner**, **parasitic_global**, **path_group**, **path_ignored_block**, **placement_spacing_rule**, **port_pin_noise_margin**, **power_clock_scaling\f, power_derate\f, pvt, pvt_dynamic, routing_rule, scaling_lib_groups, \fscan_chain, size_only, skew_group, target_library_subset,timing_derate, via_ladder_commands and voltage_area_rule.**

By default, all types of data are included in the output.

You cannot specify this option with the **-include** option.

-format dc | icc | icc2| pt

Specifies the format used to write the script files.

Valid values are

- icc2 (the default)
  The script files uses IC Compiler II command syntax.
- dc
  The script files uses Design Compiler command syntax.
- icc
  The script files uses IC Compiler command syntax.
- pt
  The script files use PrimeTime command syntax.

-output ***dir_name***

Specifies the name of the directory in which to write the script files.

The default is ./wscript.

-force

Overwrites the existing script files.

By default, the command fails if the script file already exists.

-nosplit

Prevents the splitting of command lines in the script when column fields overflow. This is most useful for doing comparisons with existing scripts or for post-processing the script.

-compress gzip

Specifies the method used to compress the generated script files.

By default, the command writes the script files as simple text files and does not compress them.

DESCRIPTION

The **write_script** command writes Tcl scripts to recreate the attributes on the current design.

By default, the **write_script** command writes command scripts to a directory named ./wscript.

This command writes multiple files to the specified directory. It writes mode-specific information to files named mode_*{mode_name}*.tcl and corner-specific information to files named corner_*{corner_name}*.tcl. It writes information that is not dependent on the mode or corner to a file named design.tcl. Clock tree synthesis options are written into a file called cts.tcl. It also writes a file named top.tcl that sources all the other script files. You can source the top.tcl script to re-create the data on the current design.

This command does not support user-defined attributes; it supports only the attributes and objects created by the following commands:

create_clock
create_generated_clock
create_routing_rule
create_voltage_area_rule
define_scaling_lib_group
group_path
set_case_analysis
set_clock_gating_check
set_clock_groups
set_clock_latency
set_clock_sense
set_clock_transition
set_clock_uncertainty
set_clock_tree_options
set_scenario_status
set_disable_timing
set_dont_touch
set_dont_touch_network
set_drive
set_driving_cell
set_extraction_options
set_false_path
set_ideal_network
set_ignored_layers
set_input_delay
set_input_transition
set_input_units

set_lib_cell_purpose
set_load
set_max_capacitance
set_max_delay
set_max_transition
set_min_capacitance
set_min_delay
set_multisource_clock_subtree_options
set_multicycle_path
set_operating_conditions
set_output_delay
set_output_units
set_placement_spacing_label
set_placement_spacing_rule
set_power_clock_scaling
set_power_derate
set_propagated_clock
set_routing_rule
set_size_only
set_temperature
set_timing_derate
set_voltage
set_aocvm_coefficient

Multicorner-Multimode Support

By default, this command works on all scenarios. To specify different scenarios, use the **-corners** and **-modes** options.

EXAMPLES

The following example writes the settings on the current design to the test_scr directory.

```
prompt> write_script -output test_scr
```

SEE ALSO

current_design(2)
reset_design(2)
write_sdc(2)

write_sdc

Writes the design constraints in Synopsys Design Constraints (SDC) format.

SYNTAX

status **write_sdc**
-output *file_name*
[-nosplit]
[-version *sdc_version*]
[-mode *mode*]
[-corner *corner*]
[-scenario *scenario*]
[-compress gzip]
[-include *include_list*]
[-exclude *exclude_list*]

Data Types

file_name string
sdc_version string
mode string
corner string
scenario string
include_list list
exclude_list list

ARGUMENTS

-output *file_name*

Specifies the name of the generated SDC file.

-nosplit

Prevents the splitting of command lines in the generated SDC file. This is most useful for doing comparisons with existing SDC files with the UNIX **diff** command or for post-processing the script.

-version *sdc_version*

Specifies the SDC version used for the generated SDC file.

Valid values are 1.7, 1.8, 1.9, 2.0, 2.1, and **latest** (the default).

-mode *mode*

Specifies the mode to write.

By default, the current mode is written.

-corner ***corner***

Specifies the corner to write.

By default, the current corner is written.

-scenario ***scenario***

Specifies the scenario to write.

By default, the scenario (if any) of the specified mode and corner is written. It is an error to use this option with the **-mode** or **-corner** options.

-compress gzip

Specifies the method used to compress the generated SDC file.

By default, the command writes the SDC file as a simple text file and does not compress it.

-include ***include_list***

Specifies the types of data to include in the output.

Valid values are **case_analysis**, **clock**, **clock_gating_check**, **clock_group**, **clock_latency**, **clock_sense**, **clock_transition**, **clock_uncertainty**, **data_check**, **data_check_ignore**, **disable_timing**, **drive_info**, **edrc**, **exception**, **file_line_info**, **ideal_network**, **input_delay**, **load**, **max_time_borrow**, **output_delay**, **path_group**, **pvt**, and **timing_derate**.

By default, all types of data are included.

This option cannot be specified with the **-exclude** option.

-exclude ***exclude_list***

Specifies the types of data to exclude from the output.

Valid values are **case_analysis**, **clock**, **clock_gating_check**, **clock_group**, **clock_latency**, **clock_sense**, **clock_transition**, **clock_uncertainty**, **data_check**, **data_check_ignore**, **disable_timing**, **drive_info**, **edrc**, **exception**, **file_line_info**, **ideal_network**, **input_delay**, **load**, **max_time_borrow**, **output_delay**, **path_group**, **pvt**, and **timing_derate**.

By default, all types of data are included.

This option cannot be specified with the **-include** option.

DESCRIPTION

The **write_sdc** command writes out a script file in Synopsys Design Constraints (SDC) format. This script contains commands that can be used in the Fusion Compiler, IC Compiler II, PrimeTime, Design Compiler, and IC Compiler tools. The SDC format is also licensed by external vendors through the TAP-in program. SDC-formatted script files are read into the Fusion Compiler, IC Compiler II, PrimeTime, Design Compiler, and IC Compiler tools by using the **read_sdc** command.

By default, the script is written as a simple text file in the latest SDC version. To write an earlier SDC version, use the **-version** option. To write a compressed SDC file, use the **-compress** option.

SDC is a subset of the commands already supported by the Fusion Compiler, IC Compiler II, PrimeTime, Design Compiler, and IC Compiler tools. The **write_sdc** command writes the following commands (those added for versions 1.7 and later are noted):

General Purpose Commands:

- expr
- list
- set

Object Access Functions:

- all_clocks
- all_inputs
- all_outputs
- all_registers (1.7)
- current_design
- current_instance
- get_cells
- get_clocks
- get_libs
- get_lib_cells
- get_lib_pins
- get_nets
- get_pins
- get_ports
- set_hierarchy_separator

Basic Timing Assertions:

- create_clock
- create_generated_clock (1.3)
- group_path (1.7)
- set_clock_gating_check
- set_clock_groups (1.7)
- set_clock_latency
- set_clock_sense (1.7)
- set_clock_transition
- set_clock_uncertainty
- set_false_path
- set_ideal_latency (1.7)
- set_ideal_transition (1.7)
- set_input_delay
- set_max_delay
- set_min_delay
- set_multicycle_path
- set_output_delay
- set_propagated_clock

New options to existing supported commands:

- create_clock -add (1.4)
- create_generated_clock -add
- create_generated_clock -master_clock
- create_generated_clock -combinational (1.7)

Secondary Assertions:

- set_disable_timing
- set_max_time_borrow

Environment Assertions:

create_voltage_area
set_case_analysis
set_drive
set_driving_cell
set_fanout_load
set_input_transition
set_ideal_network (1.7)
set_level_shifter_strategy
set_level_shifter_threshold
set_load
set_logic_dc
set_logic_one
set_logic_zero
set_max_area
set_max_capacitance
set_max_dynamic_power
set_max_leakage_power
set_max_transition
set_min_capacitance
set_min_fanout
set_min_porosity
set_operating_conditions
set_port_fanout_number
set_voltage (1.8)
set_wire_load_min_block_size
set_wire_load_mode
set_wire_load_model
set_wire_load_selection_group

Like the **write_script** command, the **write_sdc** command writes out commands relative to the top level of the design, regardless of the current instance. SDC files written by the **write_sdc** command must be read in from the top level of the design.

For a complete guide to using SDC with Synopsys tools, see the *Using the Synopsys Design Constraints Format Application Note*, which is available in SolvNet at https://solvnet.synopsys.com.

The usage of some of the supported commands is restricted when reading SDC. In some cases, some options are not allowed. The **write_sdc** command supports the restricted usage by restricting what is written.

When the hierarchy has been partially flattened, embedded hierarchy separators can make names ambiguous. It is not clear which hierarchy separator characters are part of the name, and which are real separators. You can make hierarchical names unambiguous by using the **set_hierarchy_separator** SDC command or the **-hsc** option, which is available on the **get_cells**, **get_lib_cells**, **get_lib_pins**, **get_nets**, and **get_pins** SDC object access commands. It is considered a best practice to write SDC files that contain names that are not ambiguous.

Multicorner-Multimode Support

By default, this command works on the current scenario. To specify a different scenario, use the **-scenario** option or the **-corner** and **-mode** options.

EXAMPLES

The following command writes the SDC file to a file named top.sdc:

```
prompt> write_sdc -output top.sdc
```

1

SEE ALSO

read_sdc(2)
write_script(2)

write_shadow_eco

Writes out one or more Tcl scripts that can be used to re-create feedthroughs created by **place_pins** or **push_down_objects**.

SYNTAX

status **write_shadow_eco**
[-reporting_style design_based | block_based]
[-command_style icc2 | dc]
[-output *path_name*]
[-set_shadow_status]
[-disable_undo]
[-nets *list_of_nets*]
[-cells *list_of_cells*]
[-latest_rounds *list_of_integers*]
[-self]

Data Types

path_name string
list_of_cells list

ARGUMENTS

-reporting_style design_based | block_based

Specifies how the shadow scripts should be created.

- If the style is **design_based**, the tool writes out one script.
- If the style is **block_based**, the tool writes out a set of shadow scripts for the top block and some or all the child physical blocks under top. The script for the top block contains Tcl commands to create feedthrough nets and make the correct netlist update to connect those feedthrough nets to corresponding ports or pins. The scripts for the child physical blocks contain the similar Tcl commands. Each script only makes the netlist update within the corresponding top block or the child block and no netlist update involving upper or lower physical hierarchies.

  The top block is always enabled for the shadow script creation unless the **set_editability** command explicitly disables it.

  The child physical blocks whose shadow scripts are to be created are those child physical blocks that have feedthroughs created, and at the same time specified by the **-cells** option and also enabled by **set_editability** command. If the **-cells** option is not used, then they are decided by the **set_editability** command. By default, all child physical blocks with feedthroughs created have the corresponding shadow scripts created.

The tool writes the scripts under the path specified by the **-output** option.

The default is **design_based**.

-command_style icc2 | dc

Specifies the command style used in the output Tcl script. If the command style is **icc2**, the tool writes IC Compiler II commands. If the command style is **dc**, the tool writes Design Compiler commands.

If the command style is **dc**, note that the Verilog reader tools in the Design Compiler and IC Compiler II tools can behave slightly differently when processing nets that connect to the ports of Verilog modules. In the IC Compiler II tool, the Verilog reader tries to match the net name to the Verilog module port name that the net connects to. However, if the net is declared as a wire with a name other than the name of the port it connects to, the Verilog reader changes the net name to match the port name. For any feedthroughs created on such a net, the **write_shadow_eco** command writes out the changed net name instead of the original net name.

On the other hand, the Design Compiler Verilog reader does not make the net name changes. Therefore, if Design Compiler reads in the original Verilog netlist, then Design Compiler reads in the ECO scripts created by **write_shadow_eco** in IC Compiler II, there might be errors. Design Compiler might report that some nets do not exist because of the net name changes.

To avoid such possible errors, you can either modify the original Verilog netlist to avoid using different names between nets and the module ports they connect to, or in the Design Compiler session, use a Verilog netlist created by IC Compiler II before creating the feedthroughs with the output of the **write_shadow_eco** command.

The default is **icc2**.

-output *path_name*

Specifies the output path in which to write the scripts. If the **-reporting_style** option is omitted or specified as **design_based**, and the *path_name* is an existing directory, the output script name is the same as the current design name with a ".script" extension name. The script is saved under the specified directory.

If the **-reporting_style** option is omitted or specified as **design_based**, and the *path_name* is an existing file, the output script overwrites the existing file.

If the **-reporting_style** option is omitted or specified as **design_based**, and the *path_name* is not an existing file or an existing directory, then *path_name* is treated as the full or relative path to the output script.

If the **-reporting_style** option is specified as **block_based**, and *path_name* is an existing file, then the output scripts are saved under the same directory as the existing file. The file names of the scripts are determined by the **-reporting_style** that is specified.

If the **-reporting_style** option is specified as **block_based**, and *path_name* is not an existing file, it is then treated as the full or relative path to the output directory, under which the output scripts are saved.

If this option is missing, the default output directory is the current working directory.

-set_shadow_status

If the option **-command_style** is specified as **icc2**, then setting shadow status commands are included in the output scripts.

-disable_undo

If the option **-command_style** is specified as **icc2**, then disabling undo commands are included in the output scripts.

-nets *list_of_nets*

Specifies a list of nets for which to create shadow eco scripts.

-cells *list_of_cells*

Specifies a list of cells for which to create shadow eco scripts. This option is in effect only when the option **-reporting_style** is specified as **block_based**. If the **-reporting_style** option is specified as **design_based** or omitted, then this option has no effect and is silently ignored. The child blocks that have corresponding shadow scripts are those have feedthroughs created, and at the same time specified by this option and also enabled by **set_editability** command.

If the **-cells** option is not specified, then the cells whose shadow scripts are to be created are decided by the setting of the command **set_editability**.

Note, the current top block can also be enabled or disabled for shadow script creation by the setting of **set_editability** command.

-latest_rounds ***list_of_integers***

Specifies the latest steps of **place_pins** or **push_down_objects** in which the shadow objects were created. The integers must be equal to or larger than 1. Number 1 means the latest **place_pins** or **push_down_objects** in which shadow objects were created. Number 2 means the latest but 1 (i.e. 2nd from current). And so on and so forth. If this option is used, then only shadow objects created during the specified **place_pins** or **push_down_objects** are regarded as shadow by **write_shadow_eco**. All others are regarded as original objects, even if they have shadow status *pushed_down* or *copied_down*.

In addition, to be able to recognize and honor this option, the app option **plan.pins.enable_feedthrough_timestamps** must be set true when **place_pins** or **push_down_objects** are called.

-self

Specifies that the shadow script is created for the top design.

DESCRIPTION

This command writes out one or more Tcl script that can be used to duplicate the exact same feedthrough pins, ports and nets that are created by the commands **place_pins** or **push_down_objects**, which have pushed_down or copied_down shadow attributes. When the scripts are applied to the netlist before feedthrough creation, the same feedthroughs are re-created.

The input to the command is a feedthrough created design. Additionally, it can also have buffers inserted by **add_buffer_on_route** or **add_buffers**. The command **write_shadow_eco** analyzes the netlist to identify the following objects:

- feedthrough pins, ports and nets
- any standard cells with pushed_down or copied_down shadow status
- buffers with pushed_down or copied_down shadow status
- buffers inserted by **add_buffer_on_route** or **add_buffers** on feedthrough nets
- buffers that can be 100% determined as being inserted on feedthrough nets after feedthrough creation

All of the above objects are regarded as shadow objects. The output shadow ECO scripts should re-create them and connect them correctly when applied to the design before the feedthrough creation.

In addition, the shadow ECO scripts creation can be customized by the following two app options:

- plan.shadow.exclude_non_shadow_buffers
- plan.shadow.exclude_non_hierarchical_objects

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example writes out the feedthrough creation script.

```
prompt> write_shadow_eco
```

SEE ALSO

place_pins(2)
push_down_objects(2)
plan.pins.enable_feedthrough_timestamps(3)
plan.shadow.exclude_non_shadow_buffers(3)
plan.shadow.exclude_non_hierarchical_objects(3)

write_spare_ports_eco

generate an script file that will create dummy FT ports

SYNTAX

string **write_spare_ports_eco**
[-dir *dir_name*]
[-name *port_name*]
[-direction *direction*]
[-bits *bits*]
cell_list

Data Types

dir_name string
port_name string
direction string
bits int
cell_list list

ARGUMENTS

-dir *dir_name*

Specifies the directory to output the eco files. The default directory is *create_spare_ports*. One eco file will be generated for each block. User is supposed to source the top block script, which will call the scripts for other blocks.

-name *port_name*

Specifies the names of the ports to be created. The default name is *dummyFT*.

-direction *direction*

Specifies the direction of the ports to be created. The direction contain {in | out | inout}, this is required option.

-bits *bits*

Specifies the sequence number of created ports. Start index and end index, must be non-negative value.

cell_list

Specifies the cells to create FT ports scripts.

DESCRIPTION

The **write_spare_ports_eco** command generates a script that contains the Tcl commands to that will create dummy FT ports base on customer input.

EXAMPLES

The following example writes out script for cell "U*" instances, with name Test_FT.

prompt> **write_spare_ports_eco [get_cells U*] -direction in -dir test_path -bits {0 4} -name Test_FT**

SEE ALSO

write_push_down_eco(2)

write_split_net_eco

Generate an eco script that can move the branch-out of a multi-fanout net up to top level.

SYNTAX

string **write_split_net_eco**
[-filename *file_name*]
[-nets \fnet_collection]

Data Types

file_name string
net_collection list

ARGUMENTS

-filename *file_name*

Specifies the output eco file name. If not file name not specified, the content of the eco file will be returned to the console.

-nets *net_collection*

Specifies the nets to split. If not specified, all the multi-fanout interface nets in the design will be splited. Note that a net must connect to multiple physical block pins to qualify for split. A net not connect to physical block pins will not be split. Once a net qualifies for split, then the branch-out will be moved as up as it can: the branch-out will be continued moved up, if it is driven by a pin of higher level, either it is a physical level or just a logic wrapper.

DESCRIPTION

This command will generate an eco script that can move the branch-out of a multi-fanout net up to top level.

EXAMPLES

prompt> **write_split_net_eco -filename split.tcl**
prompt> **source split.tcl**

write_taps

Writes existing taps into a text file for later reuse.

SYNTAX

status **write_taps**
-file *file_name*
[*taps*]

Data Types

string *file_name*
list *taps*

ARGUMENTS

-file *file_name*

Specifies the name of the file to which the tap points are written.

taps

Specifies the tap points to be written. The tap points can be specified as name patterns or collections of taps.

DESCRIPTION

The command writes a file containing the location and associated supply net for each inserted tap, which can be read into a later session with the **create_taps -import file** command.

The tap points written to the file can be limited to specific types by using the filter option with the **get_taps** command. For example, you can capture the set of tap points that were inserted in *top_pg* type.

The following is the format of the output file:

supply_net_name layer_number X_coordinate Y_coordinate [R(R_value)] [L(L_value)] [C(C_value)]

Lumped RLC values will be written to the tap file if they are specified for the taps. The units are Ohm for resistance, Henry for inductance, and Farad for capacitance. In the following example, no lumped RLC value is written to the tap file:

VDDY 39 746.800 469.400
VDDY 39 0.200 469.400
VDDY 39 0.200 548.600

```
VDDY 39 0.200 627.800
VSS 39 746.800 51.800
VSS 39 0.200 51.800
VSS 39 746.800 131.000
VSS 39 746.800 210.200
VSS 39 746.800 368.600
```

In the following example, the lumped RLC value is written to the tap file:

```
VDDY 39 746.800 469.400 R(0.001) L(1.0e-9) C(2.0e-12)
VDDY 39 0.200 469.400
VDDY 39 0.200 548.600
VDDY 39 0.200 627.800
VSS 39 746.800 51.800
VSS 39 0.200 51.800
VSS 39 746.800 131.000
VSS 39 746.800 210.200
VSS 39 746.800 368.600
```

Multicorner-Multimode Support

This command has no dependency on scenario-specific, mode-specific, or corner-specific information.

EXAMPLES

The following example writes a file containing the taps that were inserted via the lib_cell reference.

```
prompt> write_taps [get_taps -filter type {libcell_pg}] -file ./MyTapFile
```

SEE ALSO

create_taps(2)
remove_taps(2)
report_taps(2)
get_taps(2)

write_tech_file

Write a technology file from the current library.

SYNTAX

int **write_tech_file**
[-library *lib_name*]
file_name

ARGUMENTS

-library *lib_name*

Library from which the technology file to be written, default is current lib.

file_name

The output technology file name.

RETURN VALUE

Returns 1 if the technology file is successfully write out, and 0 if not.

DESCRIPTION

The **write_tech_file** command writes out a technology file from the user specified library. If there is no library specified, the default is current library.

EXAMPLES

The following example dump out a technology file "tcb90g.tf" from the current library.

prompt> **write_tech_file tcbn90g.tf**

SEE ALSO

create_lib(2)
read_tech_file(2)

write_test_model

Writes a test model file

SYNTAX

status **write_test_model**
-output *filename*

Data Types

filename string

ARGUMENTS

-output

File name the protocol will be written to

-format ctl | icl | pdl

Specifies the format of the test model file. By default, the model is assumed to be in .ctl format.

When **icl** is specified, the CTL test model information and an interface-only representation of the design are written out to a file. This file describe the ports for connections to Server/Subserver and TDR registers.

DESCRIPTION

The write_test_model command writes test model associated to the current design.

The test model files represent the test behavior of designs. The attributes that are represented include all DFT signals and their purpose, the test timing, and the sequence of events needed to shift test vectors in and out of the design. Test models are read from disk using the read_test_model.

Note that for standard IEEE1838, TAP port specifications inside Internal statment of testmode will adhere the following points.

- **Primary Tap:** User data will contain both "IEEE1149.1" and "IEEE1838" standards to define the port as Primary TAP. Last user data will contain the name of the tap set.

  For example, "tdo" { DataType ScanDataOut User "IEEE1149.1" User "IEEE1838" User "PTAP set name" { ScanDataType Internal; } }

- **Secondary Tap:** User data will contain only "IEEE1838" standard to define the port as secondary TAP. Last user data will contain the name of the die to which the port belongs,

  For example, "myStapTdo" { DataType ScanDataOut User "IEEE1838" User "STAP set name" { ScanDataType Internal; } }

EXAMPLES

prompt> write_test_model -output myModel.ctl

File name: myModel.ctl

1

prompt> write_test_model -format icl -output 1687.icl -test_mode ieee_1687

File name: 1687.icl

Information: ICL file '1687.icl' generated successfully.

1

SEE ALSO

read_test_model(2)

write_test_protocol

Writes a STIL test protocol file.

SYNTAX

status **write_test_protocol**
-output *filename*
[-test_mode *modename*]

Data Types

filename string
modename string

ARGUMENTS

-output

File name the protocol will be written to

-test_mode

test mode the protocol is extracted from

DESCRIPTION

The write_test_protocol command outputs the STIL test protocol from memory to disk in text format. This allows you to customize the test protocol. The scan test protocol formally defines the process by which a design is tested, so the sequence of scan-in vectors, parallel vectors, and scan-out vectors is performed for each test pattern. The protocol defined for a design is used to drive scan test and design rule checking.

EXAMPLES

```
prompt> write_test_protocol -test_mode ScanCompression_mode -output my.spf
```

SEE ALSO

create_test_protocol(2)

write_topology_constraints

Generate Tcl statements to add and/or reset multicycle constraints for topology plan objects

SYNTAX

int **write_topology_constraints**
[-filename *filename*]
[-include_reset]
[-only_reset]
[-overwrite]
[-use_object_lookup]
[topology plans]

Data Types

filename string
topology plans collection of topology plans

ARGUMENTS

-filename filename

Specifies the name for the Tcl output filename to write the multicycle constraints. Use stdout as the filename to write to the console instead.

-include_reset

Include *reset_paths* statements in the output to undo / remove the multicycle path constraints set by this command.

-only_reset

Only include the *reset_paths* statements - do not include any multicycle path statements. This option implies *-include_reset*.

-overwrite

Overwrite any files that may already exist with the given filename.

-use_object_lookup

When specified, include Tcl calls to **get_pins** or **get_ports** (as appropriate) when passing objects to the **set_multicycle_path** or **reset_paths** commands. The default is just to pass the object name and let the command determine the object(s) being referenced.

topology plans

Specifies a collection (or list of names) of topology plans to be included in the report. The default is all topology plans belonging

to the current block.

DESCRIPTION

This command creates Tcl statements of the form: set_multicycle_path -setup <count> -from <object> [-through <object>] -to <object>. and set_multicycle_path -hold <count-1> -from <object> [-through <object>] -to <object>.

for any stages of a given topology crossing through any virtual sequential repeaters (registers) that do not have any netlist implementation or do not have associated netlist objects. Timing paths default to values of "-setup 1" and "-hold 0", which is equivalent to crossing through zero virtual registers, and therefore, if all sequential topology repeaters of a given topology plan have associated std cell objects, then no statements will be generated for that topology plan. That is, statements of the form "-setup 1" and "-hold 0" will not be skipped since they are assumed by default.

The generated statements can then by used to apply suitable multicycle path constraints to the nets / objects of a topology plan when used with **report_timing**.

The reset options generate statements of a similar form, but using "reset_paths" instead of "set_multicycle_path" as the command.

Note: if the first/last object of an applicable multicycle path is a sequential element (register), then the first/last part of the path statement will be -from <clkObject> / -to <clkObject>. However, if the object is a port or block pin (non-sequential object), the first/last parts of the path statement will use "-through" instead of "-from" and "-to".

SEE ALSO

create_topology_nodes(2)
create_topology_edges(2)
create_topology_plans(2)
create_topology_repeaters(2)
report_timing(2)
report_topology_plans(2)

write_topology_plans

Writes topology plan information in Tcl format for the current design.

SYNTAX

int **write_topology_plans**
[-log]
[-include_objects]
[-current_block]
[*topology_plan_list*]
[-filename *filename*]

Data Types

topology_plan_list collection
filename string

ARGUMENTS

-log

Writes out topology plan information to the console and log file.

-include_objects

Includes Tcl commands to create bundles and supernets in the output.

-current_block

Only writes out topology plans for current block.

-filename *filename*

Specifies the file name for the topology plans output file.

topology_plan_list

Specifies a list of topology plans to write out. The list can contain topology plan names, patterns, or collections. A collection can be specified by using the **get_topology_plans** command.

DESCRIPTION

The **write_topology_plans** command generates a script that contains the Tcl commands to re-create the topology plans in the design. The script includes the topology plan name and topology edges and topology nodes. If *topology_plan_list* is specified, the command writes only the specified topology plans. If *topology_plan_list* is not specified, the command writes out all topology plans in the design.

EXAMPLES

The following example writes out information for the topology plan named "topo_5".

```
prompt> write_topology_plans topo_5 -include_objects -log
# icc2 Q-2019.12 Oct 1, 2019
# design: GPU_TOP_DG
# Fri Oct 18 13:37:16 2019

set _curr_ [current_block]

### NER (per block)

current_block GPU_TOP_DG

remove_net_estimation_rules -reset default -quiet

remove_net_estimation_rules -reset rule1 -quiet
set_net_estimation_rule rule1 -parameter "layer" -horizontal_value "M4" -vertical_value "M5"

### plan name: topo_5
current_block GPU_TOP_DG
if {[sizeof_collection [get_topology_plan topo_5 -quiet]] != 0} { remove_topology_plan topo_5 }

create_bundle -name bundle_B2A_08 [get_nets { TOP_MVECO_n8539 TOP_MVECO_n8540 TOP_MVECO_n8541 TOP_MVECC

set _objLst_ [get_bundles { bundle_B2A_08 }]
create_topology_plan -name topo_5 -objects $_objLst_ -net_estimation_rule rule1 -allow_feedthrough select_on -allow_feedthrou

create_topology_node -plan topo_5 -name node1 -objects [get_cells { A }] -constraint_type side -side W -start 190.000 -end 200.0(
create_topology_node -plan topo_5 -name node3 -objects [get_cells { E }] -constraint_type edge -edge {4}
create_topology_node -plan topo_5 -name node4 -objects [get_cells { E }] -constraint_type edge -edge {2}
create_topology_node -plan topo_5 -name node2 -objects [get_cells { B }] -constraint_type edge -edge {4} -start 120.000 -range 1(

create_topology_edge -plan topo_5 -name edge1 -nodes {topo_5/node2 topo_5/node3}
create_topology_edge -plan topo_5 -name edge1_1 -nodes {topo_5/node3 topo_5/node4}
create_topology_edge -plan topo_5 -name edge1_2 -nodes {topo_5/node4 topo_5/node1}

current_block $_curr_
1
```

SEE ALSO

create_topology_node(2)

create_topology_plan(2)
get_topology_nodes(2)
get_topology_plans(2)
remove_topology_nodes(2)
remove_topology_plans(2)
report_topology_nodes(2)

write_topology_report

Writes a report summarizing topology repeater status for implementation.

SYNTAX

int **write_topology_report**
[-filename *filename*]
[-compress]
[-no_referenced_plans]
[-overwrite]
[-verbose]
[topology plans]

Data Types

filename string
topology plans collection of topology plans

ARGUMENTS

-filename filename

Specifies the name for the XML report to be written. If not specified, default is to write to the console.

-compress

Produce a .gz compressed filed

-no_referenced_plans

By default, any plans not in the list of specified plans, but referenced by one of the specified plans using a reference node are added to the list of plans included in the report. This option prevents referenced plans from being added to the list of reported plans.

-overwrite

Overwrite any files that may already exist with the given filename

-verbose

Display extra information during processing

topology plans

Specifies a collection (or list of names) of topology plans to be included in the report. The default is all topology plans belonging to the current block.

DESCRIPTION

This command creates a report detailing the specified topology plans, including attributes, node, edge and repeater information. The included information is somewhat redundant, but is intended to make it easy for the user to find any required data to make it easy to implement registers in RTL.

SEE ALSO

create_topology_nodes(2)
create_topology_edges(2)
create_topology_plans(2)
create_topology_repeaters(2)
report_topology_plans(2)

write_verilog

Writes a Verilog description of the current design.

SYNTAX

string **write_verilog**
[-compress *method*]
[-top_module_first]
[-hierarchy *scope*]
[-split_bus]
[-include *include_list*]
[-exclude *exclude_list*]
[-force_reference *blocks_and_lib_cells*]
[-force_no_reference *blocks_and_lib_cells*]
[-switch_view_list *views*]
[-only_master_variant]
[-shell_only]
[-module_list]
[-mask_shifted_suffix_without_constraint *suffix_string*]
[-rename_cell *cell_renaming_files*]
verilog_file_name

Data Types

method string
scope string
include_list list
exclude_list list
blocks_and_lib_cells list
views list
suffix_string string
verilog_file_name string

ARGUMENTS

-compress *method*

Specifies file compression format. The only format currently supported is **gzip**. A .gz extension is appended to the file name if no extension is specified. By default, the Verilog file is not compressed.

-top_module_first

Writes the modules hierarchically top-down, so that the top module appears first in the output file. By default, modules are written hierarchically bottom-up, with the top module last in the file.

-hierarchy *scope*

Writes only the modules in the specified scope. The default *scope* value is **design**. These are the valid settings for the *scope*:

```
top         Writes only the top module in the design.
design      Writes the design, stopping at its design boundary.
design_lib  Writes the design, stopping at its design library boundary.
all         Writes the design, including modules in reference libraries.
```

-split_bus

Writes bus ports and nets as individual bits. The bus port and net names are treated as scalar names and escaped.

For example, if you have the following module definition:

```
module top (out, in);
  output [3:0] out;
  input in;
  wire [3:0] n;
  ...
endmodule
```

It is written as follows:

```
module top(\out[3] , \out[2] , \out[1] , \out[0] , in);
  output \out[3] ;
  output \out[2] ;
  output \out[1] ;
  output \out[0] ;
  input in;
  wire \n[3] ;
  wire \n[2] ;
  wire \n[1] ;
  wire \n[0] ;
  ...
endmodule
```

-include *include_list*

Specifies a list constructs and objects to include in the output. When this option is used, the Verilog contains the basic logic netlist and only the types of constructs and objects specified with this option. The valid values are:

```
all
all_physical_cells
analog_pg
corner_cells
cover_cells
diode_cells
empty_modules
end_cap_cells
feedthrough_cells
filler_cells
flip_chip_driver_cells
flip_chip_pad_cells
leaf_module_declarations
pad_cells
pad_spacer_cells
pg_netlist
pg_objects
```

physical_only_cells
scalar_wire_declarations
shadow_netlist
spare_cells
supply_statements
unconnected_ports
user_pg
well_tap_cells

For a description of each construct and object type, see the DESCRIPTION section. The default behavior is to include everything except for scalar wire declarations, leaf module declarations, PG objects, and connections to unconnected ports.

-exclude *exclude_list*

Specifies a list of constructs and objects to exclude from the output. When this option is used, the Verilog contains everything except for the types of constructs and objects specified with this option.

The valid values are:

all_physical_cells
analog_pg
corner_cells
cover_cells
diode_cells
empty_modules
end_cap_cells
feedthrough_cells
filler_cells
flip_chip_driver_cells
flip_chip_pad_cells
leaf_module_declarations
pad_cells
pad_spacer_cells
pg_netlist
pg_objects
physical_only_cells
scalar_wire_declarations
shadow_netlist
spare_cells
supply_statements
unconnected_ports
well_tap_cells

-force_reference *blocks_and_lib_cells*

Specifies blocks and lib_cells to write to the Verilog file. The *blocks_and_lib_cells* argument is either a collection or a name list of blocks and lib_cells. Instances of the block or lib_cell are written regardless of the design types specified by the **-include** or **-exclude** option. The module declaration of the block is written. The module declaration of the lib_cell is also written if you use the **-include leaf_module_declarations** option.

The forced blocks and lib_cells are not written if they are outside the scope specified by the **-hierarchy** option.

-force_no_reference *blocks_and_lib_cells*

Specifies blocks and lib_cells to omit from the Verilog file. The *block_and_lib_cells* argument is either a collection or name list of blocks and lib_cells. Instances of the block or lib_cell are omitted regardless of the design types specified by the **-include** or **-exclude** option. The module declaration of the block is omitted. The module declaration of the lib_cell is also omitted, even if you use the **-include leaf_module_declarations** option.

The **-force_reference** and **-force_no_reference** options may be used simultaneously. However if a block or lib_cell is listed in both options, it is omitted because the **-force_no_reference** option has precedence.

-switch_view_list

Specifies the reference block views for hierarchical block instances to be opened. The view at the head of the list is in higher priority.

-only_master_variant

Indicates that all the instances of variants are replaced by instances of the masters.

verilog_file_name

Specifies the name of the Verilog file to be written. Any existing file of the same name is overwritten. If the name ends with the .gz extension, it is compressed using the gzip format.

-shell_only

Writes a Verilog shell netlist that contains only the module definitions and ports; it does not include any cell instances. The **-shell_only** option cannot be used with the following options:

-compress
-exclude
-force_reference
-force_no_reference
-include
-module_list
-split_bus
-switch_view_list
-top_module_first

-module_list

Lists the modules to be defined in the output Verilog. Definitions for all other modules are omitted.This option has precedence over all other options in controlling which module definitions appear in the output Verilog. This option does not control the output of the module contents, such as the inclusion or exclusion of module instances.

If the **-module_list** is not specified, all modules within the specified hierarchy scope are written (see **-hierarchy** option).

The **-module_list** option cannot be used with the following options:

-hierarchy
-shell_only

-mask_shifted_suffix_without_constraint *suffix_string*

Specifies the suffix string used to generate reference cell name when double-patterning mask shifting enabled in the flow. If a cell instance has the mask_shift attribute set to none zero value, this option will write the reference name of this instance as <original_reference_name>_<suffix_string>. Without this option, the reference name of the instance with non-zero mask_shift will be the original name.

-rename_cell *cell_renaming_files*

Specifies a file that contains the mapping from original mask shifted cell names to new cell names. The renaming works for both reference library cells and design blocks.

The name mapping comprises three sections: MASKSHIFTLAYER, MASKSHIFTPATTERN and CELLMAP.

For example, to rename the cell A with mask shifting on layer M1 or/and M2, enter the following:

```
<MASKSHIFTLAYER>
M1 M2
</MASKSHIFTLAYER>
<MASKSHIFTPATTERN>
00 01 10 11
</MASKSHIFTPATTERN>
<CELLMAP>
A AM2B NONE AM1BM2B
</CELLMAP>
```

MASKSHIFTLAYER specifies the relevant metal layers, in order, on a single row.

MASKSHIFTPATTERN specifies the mask shift combinations to apply to the metal layers. The digits in each combination correspond to the metal layers listed in MASKSHIFTLAYER, respectively. For example, the 00 combination represents a shift value of 0 on layers M1 and M2; the 01 combination represents a shift value of 0 on layer M1 and 1 on layer M2; the 10 combination represents a shift value of 1 on layer M1 and 0 on layer M2; and so on.

The number of mask shift combinations is less than or equal to pow(m, n), where n is the number of layers specified in MASKSHIFTLAYER and m is the count of the shift values. For 2-mask layers, valid shift values are 0 and 1, so m is 2. In the example above, two metal layers are specified (M1 M2), so there are four mask shift combinations (00 01 10 11).

CELLMAP specifies the cell mappings that correspond to each mask shift pattern for a given cell. (Each row specifies the mappings for a single cell, so the number of mappings in each row corresponds to the number of MASKSHIFTPATTERN combinations.) The **write_verilog** command renames a cell based on the cell's mask_shift attribute and the mapping file content.

In the example above, cell A is mapped to A, AM2B, NONE, and AM1BM2B when its mask_shift attribute is set to 0 on both layers (00), set to 1 on layer M2 (01), set to 1 on layer M1 (10), and set to 1 on both layers (11), respectively. When a mask-shifted cell is not expected for a particular mask shift combination, the cell name is mapped to NONE and **write_verilog** errors out. When a mask-shifted cell is undefined, the mapping file does not contain the corresponding name mappings and **write_verilog** errors out.

By default, the original cell name is preserved in the output verilog file.

DESCRIPTION

This command writes a hierarchical Verilog netlist from the current design. The current design is automatically linked if it is not already linked. The modules are ordered bottom-up by default, or top-down if you use the **-top_module_first** option. If the design is not linked and auto-linking fails, only the modules of the current design are written.

Before you use the **write_verilog** command, to change the names of objects in the design to conform to Verilog naming conventions, use the **change_names -rules verilog** command.

When you use the **-include** option, the command writes out the basic logic netlist and only the objects specified with this option. When you use the **-exclude** option, the command writes out everything except for the objects specified with this option. If you use neither of these options, the output includes everything except scalar wire declarations, leaf module declarations, PG objects, and connections to unconnected ports. This is equivalent to specifying:

```
-exclude { scalar_wire_declarations leaf_module_declarations
        pg_objects unconnected_ports }
}
```

These are the valid values for the *include_list* and *exclude_list*:

```
GENERAL
------------------------------
```

empty_modules Module declarations for hierarchical (non-leaf) cells, which may have port declarations, but no internal module instances and net connections regardless of dangling nets in it. Both the empty module and its cell instances will be included or excluded.
leaf_module_declarations Module declarations for leaf-level cells. These module declarations contain port declarations, but no internal module instances.
scalar_wire_declarations Scalar wire declarations.
shadow_netlist Shadow netlist cells, nets, ports, and pins.
unconnected_ports Connections to unconnected ports on module instances. These connections are written out with a net named "SYNOPSYS_UNCONNECTED_" plus a unique integer.

POWER AND GROUND

analog_pg Analog power and ground nets, ports, and pin connections.
pg_netlist Power and ground nets, ports, and pin connections.
pg_objects All power and ground objects. Specifying this value is equivalent to specifying the analog_pg, pg_netlist, and supply_statements values.
supply_statements Supply1 and Supply0 declarations for power and ground nets.
user_pg Power and ground nets, ports, and pins, along with tie nets, which have **is_user_pg** attribute values of true.

CELL TYPE

all_physical_cells Physical cells in any of the below categories. Specifying this value is equivalent to specifying all of the other individual values below.
corner_cells Physical cells with design_type "corner".
cover_cells Physical cells with design_type "cover".
diode_cells Physical cells with design_type "diode".
end_cap_cells Physical cells with design_type "end_cap".
feedthrough_cells Physical cells with design_type "feedthrough".
filler_cells Physical cells with design_type "filler".
flip_chip_driver_cells Physical cells with design_type "flip_chip_driver".
flip_chip_pad_cells Physical cells with design_type "flip_chip_pad".
pad_cells Physical cells with design_type "pad".
pad_spacer_cells Physical cells with design_type "pad_spacer".
physical_only_cells Physical cells which are physical-only (i.e., cell is_physical_only attribute is true).
spare_cells Physical cells which are unconnected or only connected to constant nets.
well_tap_cells Physical cells with design_type "well_tap".

The *include_list* also allows the following value:

all All of the objects in the other categories

EXAMPLES

The following example writes a Verilog file:

```
prompt> write_verilog my_output.v
```

The following example writes a Verilog file with all constructs and objects:

```
prompt> write_verilog -include {all} my_output.v
```

The following example writes a Verilog file with everything except cover and well tap cells.

```
prompt> write_verilog -exclude {cover_cells well_tap_cells} my_output.v
```

The following example writes a Verilog file with everything except scalar wire declarations and leaf module declarations. It implicitly includes all physical cells, the PG netlist, and supply statements.

```
prompt> write_verilog -exclude \
  { scalar_wire_declarations leaf_module_declarations} my_output.v
```

The following example writes a Verilog file with everything except scalar wire declarations, leaf module declarations, and supply statements. It implicitly includes all physical cells and the PG netlist.

```
prompt> write_verilog -exclude {scalar_wire_declarations \
  leaf_module_declarations supply_statements} my_output.v
```

The following example writes a Verilog file containing the user PG netlist and associated supply statements, but omits the non-user PG netlist.

```
prompt> write_verilog -include {user_pg supply_statements} my_output.v
```

The following example writes a Verilog file containing the modules mid and bot.

```
prompt> write_verilog -module_list {mid bot} my_output.v
```

SEE ALSO

change_names(2)
read_verilog(2)

write_virtual_pad_file

Saves all existing virtual pad locations and layers to a file.

SYNTAX

status **write_virtual_pad_file**
file_name

Data Types

file_name string

ARGUMENTS

file_name

Specifies a file name into which to write all existing virtual pad locations and layers.

DESCRIPTION

This command saves all existing virtual pad locations and layers to a file.

Multicorner-Multimode Support

This command is not supported in a multi-scenario design flow.

EXAMPLES

The following example saves all existing virtual pads to a file name test.vpad.

prompt> **write_virtual_pad_file test.vpad**

SEE ALSO

analyze_power_plan(2)
read_virtual_pad_file(2)
remove_virtual_pads(2)
report_virtual_pads(2)
set_virtual_pad(2)